Final Report

Title: Information-based security protocols for ideal agents

Contract Number: FA520905P0135

AFOSR/AOARD Reference Number: AOARD-05-4017

AFOSR/AOARD Program Manager: Tae-Woo Park, Ph.D.

Period of Performance: 01 October 2004 – 01 October 2005

Submission Date: 10 February 2006

PI: dr H.P. van Ditmarsch / University of Otago, PO Box 56, Dunedin, New Zealand
CoPI: N/A

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
08 AUG 2006	**Final Report (Technical)**	**01-10-2004 to 01-10-2005**

4. TITLE AND SUBTITLE
Information-based Security Protocols for Ideal Agents

5a. CONTRACT NUMBER
FA520905P0135

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Hans Pieter van Ditmarsch

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Otago,P.O. Box 56,Dunedin 9015,New Zealand,NZ,9015

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
The US Resarch Labolatory, AOARD/AFOSR, Unit 45002, APO, AP, 96337-5002

10. SPONSOR/MONITOR'S ACRONYM(S)
AOARD/AFOSR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AOARD-054017

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The objective to model check properties of protocols for card playing agents has been met. Beyond the objectives we are pleased to report that we have made a state-of-the-art comparison of epistemic model checkers (see publication [1], below). We have paid some attention to other protocols, namely different generalizations of card playing protocols (see archival documentation [4]). We applied model checking tools to a rather different protocol (that has the form of a riddle about natural numbers), published in [2]. We did not get to the promised 'library of protocols'. The reason for that is that the computational complexity of model checking protocols is far higher than we anticipated. (For example, the most basic generalization of the 'five-hand's' protocol mentioned in [1] takes up to 5 hours to model check - and this can only be achieved by 'real hacking', a proper structural programming approach has too high complexity.) Some theoretical/technical progress would be necessary before further challenges can be undertaken. We think the project results are appreciated by the academic community. Publications [1] and [2] have already been quoted in research proposals, other publications or submissions, and have directed other researchers to use the compared model checkers.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**100**	

Objectives: The objectives of this research proposal are to build a small library of protocols that have been described in a model checking formalism and verified in a model checker; to prove or disprove the conjectured properties of card protocols mentioned above, also for more general setting of distributive systems with interdependencies among agents. Other types of security protocol that are unrelated to the protocols for information sharing in card deals will also be described, investigated, and verified. A systematic search of the literature will therefore be part of the investigation.

Status of effort: The research has been completed. Most of the work was done in the period March-August 2005. Some publications resulting from this project, and related continuing efforts, will / may appear later.

Abstract: The objective to model check properties of protocols for card playing agents has been met. Beyond the objectives we are pleased to report that we have made a state-of-the-art comparison of epistemic model checkers (see publication [1], below). We have paid some attention to other protocols, namely different generalizations of card playing protocols (see archival documentation [4]). We applied model checking tools to a rather different protocol (that has the form of a riddle about natural numbers), published in [2]. We did not get to the promised 'library of protocols'. The reason for that is that the computational complexity of model checking protocols is far higher than we anticipated. (For example, the most basic generalization of the 'five-hand's' protocol mentioned in [1] takes up to 5 hours to model check - and this can only be achieved by 'real hacking', a proper structural programming approach has too high complexity.) Some theoretical/technical progress would be necessary before further challenges can be undertaken.

We think the project results are appreciated by the academic community. Publications [1] and [2] have already been quoted in research proposals, other publications or submissions, and have directed other researchers to use the compared model checkers.

Personnel Supported:

- (principal investigator) Dr H.P. van Ditmarsch, University of Otago, Dunedin, New Zealand
- (co-investigator) Professor M.D. Atkinson, University of Otago, Dunedin, New Zealand
- (co-investigator) Professor W. van der Hoek, The University of Liverpool, United Kingdom
- (co-investigator) Associate Professor R. van der Meyden, NICTA/UNSW, Sydney, Australia
- (research assistant of principal investigator) Mr J. Ruan, University of Otago, Dunedin, New Zealand

- (student) Ms Sigrid Roehling, University of Otago, Dunedin, New Zealand
- (student) Mr Daniel Wood, University of Otago, Dunedin, New Zealand
- (student) Mr Huifeng Chen, University of Otago, Dunedin, New Zealand

Publications:

1. H.P van Ditmarsch, W. van der Hoek, R. van der Meyden, J. Ruan, *Model Checking Russian Cards*. Electronic Notes in Theoretical Computer Science 149:105-123, 2006.
2. H.P. van Ditmarsch, J. Ruan, L.C. Verbrugge, *Model Checking Sum and Product*. In: S. Zhang and R. Jarvis (editors), Proceedings of the 18th Australian Joint Conference on Artificial Intelligence (AI 2005) (LNAI 3809), pages 790-795, Springer Verlag, Berlin, 2005.
3. H.P. van Ditmarsch, *The case of the hidden hand*. Journal of Applied Non-Classical Logics. To appear.

Interactions:

(a) Participation/presentations at meetings, conferences, seminars, etc., have been:

- (Hans van Ditmarsch) Math Dept seminar, National University of Singapore, 8 Nov 2004
- (Ron van der Meyden) presentation at Software and Sensor Fusion Workshop, Griffiss Institute, Rome NY, 15 August 2005
- (Ron van der Meyden) Workshop presentation, MoChArt 2005, 27 August 2005, San Francisco
- (Hans van Ditmarsch) seminar, NICTA (National ICT Australia) Queensland, 31 January 2006

(b) I do not know of cases where knowledge resulting from this project effort is, or will be, used.

New:

(a) None.
(b) The completed **"DD Form 882, Report of Inventions and Subcontractors"** is sent separately by the University of Otago Research & Enterprise Office. (The legal authority to sign that document.)

Honors/Awards: None

Archival Documentation: The following electronic reprints have been attached to this report - with the exception of items [6] and [7], that we consider of minor interest (but are by all means available on request).

1. (publication 1, above)
2. (publication 2, above)
3. (publication 3, above) (in press, attached is the final version as accepted for publication)
4. Ron van der Meyden, AOARD Trip Report (Software and Sensor Fusion Workshop, Griffiss Institute, Rome)
5. Sigrid Roehling, *Cards Cryptography* (University of Otago, 480 project report)
6. Daniel Wood, *Cards Cryptography* (University of Otago, 480 project report)
7. Huifeng Chen, *Model Checking of Knowledge* (University of Otago, 480 project report)
8. Hans van Ditmarsch, Ji Ruan, Rineke Verbrugge, *Sum and Product in Dynamic Epistemic Logic* (University of Otago, Computer Science, Technical Report OUCS-2006-01)

Software and/or Hardware (if they are specified in the contract as part of final deliverables):

None

Cards and Cryptography

(COSC 480)

Final Report

Sigrid Roehling

Supervised by Mike Atkinson

University of Otago

submitted in partial fulfilment of the degree of

Master of Science

at the University of Otago, Dunedin,

New Zealand.

October 6, 2005

Abstract

From a deck of $a + b + c$ distinct cards agents Alice and Bob receive a and b cards, respectively, and Cathy, the attacker, receives the remaining c cards. Methods have been devised for creating announcements that Alice can make in order to inform Bob about her cards without Cathy learning any card held by either of them (Albert *et al.*, 2003). This dissertation extends the problem by requiring that the announcement not give Cathy a greater than random chance to *guess* any of Alice's or Bob's cards. Precise conditions for the announcement are given and several announcements are presented and analyzed.

Contents

Chapter 1

Introduction

1.1 Background

Public key cryptography bases its security on mathematical problems that are difficult to solve such as the discrete logarithm problem or factoring the product of two large primes. Advances in technology and new discoveries in mathematics make it more feasible to solve these problems, i.e. it becomes more feasible to break the encryption. One solution is to use larger prime numbers to raise the bar a little higher, but this also translates to more computation needed to actually use the encryption, making it more inconvenient for the user. In addition, new mathematical discoveries may suddenly provide an easy way to solve these problems and therefore render the complete algorithm useless for encryption (Wikipedia, 2005). Instead, one might look at cryptographic protocols in which discovering the secret is not only too complex given the current state of technology but actually impossible. That is, the cryptographic protocol must be developed with a computationally unlimited attacker in mind. One such approach involves the use of a random deck of cards. The general scenario is as follows: Two agents, Alice and Bob, draw a and b cards from a deck of $a + b + c$ cards, and Cathy, the attacker, receives the remaining c cards. Alice wishes to communicate her cards to Bob by making a public announcement without informing Cathy of any of her cards. The generalized problem has parameters (a, b, c) and was inspired by what van Ditmarsch (2003) has called the Russian Cards problem, which constitutes the $(3, 3, 1)$ instance and was presented at the Moscow Mathematical Olympiad in 2000.

1.2 Previous Work

Previous work on the Russian Cards problem involved using epistemic logic to describe the properties and find several solutions (van Ditmarsch, 2003). The generalized version has been investigated by Albert, Aldred, Atkinson, van Ditmarsch, and Handley (2003) with the following results: Alice can structure her announcement in many different ways, but they all correspond to an announcement of the form “I hold one of the following hands: ...”. The work of Albert *et al.* on this problem included devising precise mathematical conditions, called CA1, CA2 and CA3, corresponding to the requirements above. Using these conditions, methods of constructing “good

announcements” have been found.

Example of a Good Announcement

For the $(3, 3, 1)$ instance, suppose Alice holds 034. She could announce that she holds one of $\{012, 034, 056, 135, 246\}$. No matter what Bob holds he can infer Alice’s cards. For example, if he held 126, then he could eliminate 012, 056, 135 and 246, leaving only 034, Alice’s actual hand. And no matter what Cathy holds, she cannot infer any card of Alice or Bob. Say Cathy held card 5, then she could eliminate 056 and 135, leaving her with 012, 034 and 246 for Alice’s hand and (by considering the remaining cards) 346, 126 and 013 for Bob’s hand. Tables 1.1 and 1.2 illustrate these properties given the announcement mentioned above.

		Hands of the announcement				
		012	034	056	135	246
Possible hands held by Bob	013					X
	015					X
	024				X	
	026				X	
	035					X
	046				X	
	123			X		
	124			X		
	125		X			
	126		X			
	134			X		
	135					X
	156		X			
	234			X		
	246				X	
	256		X			
	345	X				
	346	X				
	356	X				
	456	X				

Table 1.1: Illustration of Bob’s inference of Alice’s hand. (An X marks the hand of Alice that Bob will deduce.)

1.3 New Problem

Some of the constructions proposed by Albert *et al.*, while not giving away enough information for Cathy to *determine* any card held by Alice or Bob, will result in situations where Cathy can make an *educated guess* based on the relative frequency of the cards. For example, consider again the announcement for $(3, 3, 1)$ given as

		Hands of the announcement				
		012	034	056	135	246
Possible cards held by Cathy	0				X	X
	1		X	X		X
	2		X	X	X	
	3	X		X		X
	4	X		X	X	
	5	X	X			X
	6	X	X		X	

Table 1.2: Illustration of Cathy's attempt at inferring Alice's hand. (An X marks what Cathy believes to be a possible hand of Alice.)

$\{012, 034, 056, 135, 246\}$. Let us assume that Cathy holds card 3. Then she can exclude all but 012, 056 and 246 from the announcement. Since among these card 2 occurs more often than card 1, Cathy knows that Alice is more likely to hold card 2.

This is based on two assumptions: First, the announcements that Alice makes are closed under relabeling, meaning that we can take one announcement and relabel, say, card 0 as card 1, card 1 as card 2, ..., card 6 as card 0 to get a new announcement. Thus, we can make use of the announcement above even if we do not hold any of the hands 012, 034, 056, 135 and 246. We simply pick one of these hands and relabel it so that it matches our actual hand and then apply the same relabeling to the other hands in the announcement. The second assumption is that Alice will randomly choose an announcement that fits her actual hand. It is important to note that among all announcements that Alice can make there are those in which Alice's cards occur more frequently than others and those in which her cards occur less frequently or just as frequently. However, there are more of the first kind than there are of the second (Michael Albert, personal communication). With Alice choosing an announcement at random she is more likely to pick one in which her cards occur more frequently. Therefore, not having any other information, Cathy's best bet is to assume that when a card occurs more often, it is more likely to be one of the actual cards.

This dissertation devises additional requirements for the announcement in order to eliminate the possibility of making educated guesses, examines the constructions previously proposed, and searches for methods of designing announcements that meet these requirements.

Chapter 2

Theory of New Problem

2.1 Terminology

I will use terminology previously proposed by Albert *et al.* (2003). In their terminology cards are commonly referred to as points and labeled with numbers. The set of all cards is denoted by Ω. An x-set is a set of x cards. A possible holding (or hand) of Alice is called a line. Thus, an announcement $\mathcal{L}$ consists of one or more lines. "Elimination" refers to Cathy eliminating those lines from the announcements that are impossible holdings for Alice because they contain one or more cards that Cathy herself is holding.

2.2 Revisiting CA1, CA2, and CA3

Albert *et al.* proposed three axioms that correspond to the informal requirements given in the problem description. They are restated and explained below.

2.2.1 Combinatorial Axiom 1 (CA1)

Given the announcement, Bob must be able to infer what Alice is holding.

Combinatorial Axiom 1. *For every b-set X there is at most one line in $\mathcal{L}$ that avoids X.*

In order for Bob to figure out which line of the announcement is Alice's holding, he has to eliminate lines from the announcement based on his knowledge of his own cards. For example, because cards are distinct, if Bob holds card 4, then he can eliminate all lines that contain card 4 since those cannot be a possible holding of Alice. Similarly, Bob can eliminate any other line that contains a card that he himself holds. A line in the announcement that contains none of the cards held by Bob is said to avoid Bob's hand (here denoted by b-set X). If there are two or more such lines in the announcement, then Bob is left with more than one possibility for Alice's hand and cannot state with absolute certainty which is the correct one. Therefore, there can be at most one line in the announcement that avoids Bob's hand. (Note: If we are assuming that the announcement is truthful and that Alice's hand is among the lines then there must be *exactly* one line that avoids Bob's hand.)

2.2.2 Combinatorial Axiom 2 (CA2)

Given the announcement, Cathy must not be able to infer *any* card held by Alice.

Combinatorial Axiom 2. *For every c-set X the lines in $\mathcal{L}$ avoiding X have empty intersection.*

Cathy employs the same process of eliminating lines from the announcement as Bob by looking at her own hand (denoted by c-set X). After elimination, she examines the remaining lines. If there is one card common to all these lines, then Cathy can conclude that Alice holds that card. So, there must be no card common to all remaining lines. In other words, all remaining lines taken together must have empty intersection.

2.2.3 Combinatorial Axiom 3 (CA3)

Given the announcement, Cathy must not be able to infer *any* card held by Bob.

Combinatorial Axiom 3. *For every c-set X the lines in $\mathcal{L}$ avoiding X have union consisting of all cards except those of X.*

After elimination Cathy also examines the remaining lines for information about Bob's cards. For example, if there is a card that does not occur among these lines then it is not held by Alice. It is also not held by Cathy because she has eliminated all lines containing any of her cards. Thus, it must be a card held by Bob. If, however, the remaining lines cover all possible cards, except for Cathy's, then Cathy cannot draw such conclusions about Bob's cards. In other words, the union of the remaining lines must be the set of all cards minus the set X of Cathy's cards.

2.3 New Requirements CA4 and CA5

2.3.1 Combinatorial Axiom 4 (CA4)

Given the announcement, Cathy must not have a greater than random chance of guessing any card held by Alice.

Combinatorial Axiom 4. *For every c-set X, let $D_X = \{L \in \mathcal{L} \mid L \cap X = \emptyset\}$. The number of lines n_X in D_X that contain a particular card $y \notin X$ must be independent of y.*

D_X is the subset of the announcement that does not intersect with X, i.e. it contains the lines that Cathy cannot eliminate by looking at her own hand. If among these lines any card occurs more often than another, then that card is more likely to be a holding of Alice. This gives Cathy a better than random chance of guessing one of Alice's cards correctly. In order to prevent this, all cards (except Cathy's) must occur equally often in D_X. (Note that by elimination Cathy's cards do not occur in D_X at all.)

2.3.2 Combinatorial Axiom 5 (CA5)

Given the announcement, Cathy must not have a greater than random chance of guessing any card held by Bob.

Combinatorial Axiom 5. *For every c-set X, let D_X be the same as in CA4 and let $E_X = \{\Omega - X - L \mid L \in D_X\}$. The number of sets in E_X that contain a particular card $y \notin X$ must be independent of y.*

Given the remaining lines in D_X, Cathy can construct all possible holdings for Bob by taking the complements of the lines with respect to the set of cards held by Alice and Bob $(\Omega - X)$. This produces E_X, the set of possible holdings of Bob. Similarly to CA4, all cards (except Cathy's) have to occur equally often in E_X in order to prevent Cathy from having a better than random chance of guessing one of Bob's cards correctly.

2.3.3 Link Between CA4 and CA5

Because E_X used in CA5 is constructed from D_X used in CA4, they seem to be closely related. In fact, CA4 implies CA5, and vice versa. For a proof see Chapter 4.

Chapter 3

Examples of Announcements

3.1 Short Introduction to Designs

The mathematical theory of designs which deals with collections of special subsets of a given set provides a convenient framework for studying the conditions CA1, CA2, CA3, CA4 and CA5.

> "Informally, one may define a combinatorial design to be a way of selecting subsets from a finite set in such a way that some specified conditions are satisfied." (Wallis, 1988)

For example, the set of subsets $\{\{0,1\},\{2,3,4\}\}$ over the finite set $\{0,1,2,3,4\}$ satisfies the condition that every member of the finite set occurs in exactly one subset. The special subsets will be called lines for the remainder of this dissertation. Designs with different properties have been studied extensively, but for our purposes a short introduction to t-designs will suffice. In general, t-designs have the property that any combination of t distinct cards occurs in the same number of lines. This number is referred to as the covalency λ of the design. Thus, in 2-designs (also known as balanced incomplete block designs) any pair of distinct cards occurs in the same number of lines. Similarly, in 3-designs any 3-tuple of distinct cards occurs in the same number of lines. Generally, t-designs have a feature akin to backwards compatibility in that every t-design is also a 1-design, 2-design, $\ldots$, $(t-1)$-design (Hughes, 1962). The following first three designs represent some of the announcements I have found to date that satisfy CA4 and CA5. They all appear to be 2-designs. In fact, projective planes are 2-designs by definition and binary designs are 3-designs as shown in Chapter 4. For finite field designs I merely have experimental evidence in that all the ones tested have been 2-designs. (Satisfaction of t-design properties has been evaluated using a computer program. See Appendix C for more details.) Following these is a short list of other announcements I have come across that have been analyzed in less detail.

3.2 Projective Planes

3.2.1 Properties

Projective planes have the following properties:

- All lines contain the same number of points.
- Any pair of points occurs in exactly one line.
- Any pair of lines intersects in exactly one point.

Note that the second property makes projective planes 2-designs (with $\lambda = 1$).

3.2.2 Construction

Projective planes can be constructed using fields (see Appendix B). The number of points is of the form $1+q+q^2$. However, projective planes do not exist for all values of q. All planes known to date are associated with a q that is a power of a prime and it is known that there exists a plane with $q = p^2$ for every odd prime p (Assmus and Key, 1992). No plane has been found yet with q not a prime power but the nonexistence of these has not been proven either.

3.2.3 Example

An example of a projective plane is the 7-line announcement for $(3, 3, 1)$ shown in Figure 3.1. The nodes of the graph represent the points, and the edges (including the circle) represent the lines. Representing each line $\{x, y, z\}$ as xyz, the corresponding announcement is $\{012, 034, 056, 135, 146, 236, 245\}$. Table 3.1 illustrates how this design satisfies CA4. Note that $n_X = 2$ for all possible X, i.e. it is independent of X. This is not an explicit requirement of CA4, but may be a consequence. Theorem 4 in Chapter 4 explores this in more detail. As shown in Chapter 4, CA4 implies CA5, thus projective planes also satisfy CA5.

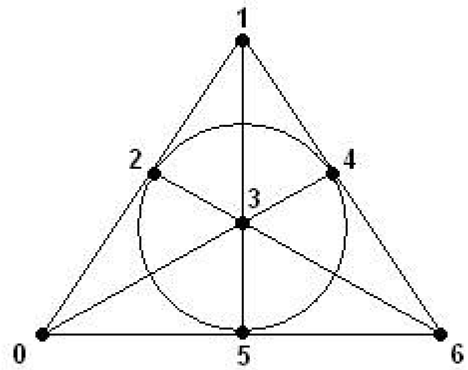

Figure 3.1: The 7-point projective plane.

3.3 Binary Designs

3.3.1 Properties

Binary designs give solutions for $a = 2^{n-1}, b = a - 1, c = 1, n \geq 3$. Here, n is the number of bits used in the construction. These designs are special because the same n

$$Announcement : \{012, 034, 056, 135, 146, 236, 245\}$$

		Occurrences							
X	D_X	0	1	2	3	4	5	6	n_X
0	{135, 146, 236, 245}		2	2	2	2	2	2	2
1	{034, 056, 236, 245}	2		2	2	2	2	2	2
2	{034, 056, 135, 146}	2	2		2	2	2	2	2
3	{012, 056, 146, 245}	2	2	2		2	2	2	2
4	{012, 056, 135, 236}	2	2	2	2		2	2	2
5	{012, 034, 146, 236}	2	2	2	2	2		2	2
6	{012, 034, 135, 245}	2	2	2	2	2	2		2

Table 3.1: Illustration of CA4 for the 7-point projective plane for $(3, 3, 1)$.

is associated with more than one instance of the (a, b, c) problem. For example, $(8, 7, 1)$ and $(8, 6, 2)$ have the same solution given by a binary design with $n = 4$. Binary designs are not only 2-designs, but also 3-designs as has been proven in Chapter 4. Whether this has anything to do with their special nature has yet to be investigated.

3.3.2 Construction

Choose the number of bits $n \geq 3$. For all $2^n - 1$ n-bit vectors $(b_1, b_2, ..., b_n)$ (except all zeros) solve the equation

$$x_1 b_1 + x_2 b_2 + ... + x_n b_n = 0$$

where $x_y = 0, 1$. Note that we are working in binary, so everything is taken modulo 2. There will be 2^{n-1} solutions to each equation, each $x_1 x_2 ... x_n$ representing a point in binary. All points gained from an equation constitute a line. This produces $2^n - 1$ lines (one per equation). For each line, compute the complement by taking all binary points that are *not* present in the line; these complements are also taken as lines. Now we have a total of $2(2^n - 1)$ lines which constitute the announcement. To generate the final announcement, replace every point in binary with its decimal representation.

3.3.3 Example

Construct a binary design with $n = 3$. Each line consists of $2^{n-1} = 4$ points. The $2^n - 1 = 7$ non-zero 3-bit vectors are

$$001, 010, 011, 100, 101, 110, 111.$$

The corresponding lines are

$$\{000, 010, 100, 110\},$$
$$\{000, 001, 100, 101\},$$
$$\{000, 011, 100, 111\},$$
$$\{000, 001, 010, 011\},$$
$$\{000, 010, 101, 111\},$$
$$\{000, 001, 110, 111\},$$
$$\{000, 011, 101, 110\}.$$

together with their complements (respectively)

$$\{001, 011, 101, 111\},$$
$$\{010, 011, 110, 111\},$$
$$\{001, 010, 101, 110\},$$
$$\{100, 101, 110, 111\},$$
$$\{001, 011, 100, 110\},$$
$$\{010, 011, 100, 101\},$$
$$\{001, 010, 100, 111\}.$$

The $2(2^n - 1) = 14$ lines in decimal are (respectively)

$$\{0, 2, 4, 6\}, \{0, 1, 4, 5\}, \{0, 3, 4, 7\}, \{0, 1, 2, 3\}, \{0, 2, 5, 7\}, \{0, 1, 6, 7\}, \{0, 3, 5, 6\},$$
$$\{1, 3, 5, 7\}, \{2, 3, 6, 7\}, \{1, 2, 5, 6\}, \{4, 5, 6, 7\}, \{1, 3, 4, 6\}, \{2, 3, 4, 5\}, \{1, 2, 4, 7\}.$$

Table 3.2 illustrates how this announcement satisfies CA4 for the (4,3,1) instance.

3.4 Finite Field Designs

3.4.1 Properties

These designs make use of the properties of finite fields. For the mathematically inclined, we construct a finite field of size p^2 and take the quadratic residues, the quadratic non-residues, and their translates to be the lines. The construction below is written for those with less detailed knowledge of mathematics, but an introduction to fields can be found in Appendix B. These designs give solutions for $a = \frac{p^2-1}{2}$, where $p \geq 3$ is prime, and $b = a, c = 1$.

3.4.2 Construction

Choose a prime $p \geq 3$. Let $T = \{a + bx \mid a, b = 0, 1, 2, ..., p - 1\}$. Arithmetic is modulo p. Find some pair of values of a and b in $f(x) = x^2 + ax + b$ where $a, b = 0, 1, 2, ..., p - 1$

$$\begin{aligned} Announcement : \{&0123, 0145, 0167, 0246, 0257, 0347, 0356,\\ &4567, 2367, 2345, 1357, 1346, 1256, 1247\} \end{aligned}$$

		Occurrences								
X	D_X	0	1	2	3	4	5	6	7	n_X
0	{4567, 2367, 2345, 1357, 1346, 1256, 1247}		4	4	4	4	4	4	4	4
1	{0246, 0257, 0347, 0356, 4567, 2367, 2345}	4		4	4	4	4	4	4	4
2	{0145, 0167, 0347, 0356, 4567, 1357, 1346}	4	4		4	4	4	4	4	4
3	{0145, 0167, 0246, 0257, 4567, 1256, 1247}	4	4	4		4	4	4	4	4
4	{0123, 0167, 0257, 0356, 2367, 1357, 1256}	4	4	4	4		4	4	4	4
5	{0123, 0167, 0246, 0347, 2367, 1346, 1247}	4	4	4	4	4		4	4	4
6	{0123, 0145, 0257, 0347, 2345, 1357, 1247}	4	4	4	4	4	4		4	4
7	{0123, 0145, 0246, 0356, 2345, 1346, 1256}	4	4	4	4	4	4	4		4

Table 3.2: Illustration of CA4 for the binary design with $n = 3$ for $(4, 3, 1)$.

such that $f(x) \neq 0$ for all $x = 0, 1, 2, ..., p-1$. Formally solve $f(x) = 0$ for x^2, i.e. note that $x^2 = -ax - b$. This is the multiplication rule, i.e. whenever we encounter x^2 we will replace it by the righthand side of the rule.
Now, find $t \in T$ such that $t, t^2, ..., t^{p^2-1}$ are all different from 1. The set of all even powers of t (including t^0) form line L_0. The set of all odd powers of t form line L_1. The lines of the announcement are constructed as follows: For every $s \in T$ add s to every member of L_0 and L_1 to generate two lines. For convenience, assign a distinct number to each polynomial in T and replace the polynomials in the lines by the respective number.

3.4.3 Example

Constructing a finite field design with $p = 3$.

$$T = \{a + bx \mid a, b = 0, 1, 2\}.$$

Arithmetic is modulo 3. The multiplication rule is $x^2 = x + 1$ since $f(x) = x^2 + 2x + 2$ is never zero for $x = 0, 1, 2$, i.e. $f(0) = 2 \bmod 3 = 2$, $f(1) = 5 \bmod 3 = 2$, and $f(2) = 10 \bmod 3 = 1$. Note that in modulo-3 arithmetic $x^2 = -2x - 2 = x + 1$. $t = 1 + 2x$ gives $L_0 = \{1, 2 + 2x, 2, 1 + x\}$ and $L_1 = \{1 + 2x, x, 2 + x, 2x\}$.

The lines of the announcement are (from L_0)

$$
\begin{array}{llll}
\{1, & 2+2x, & 2, & 1+x\}, \\
\{1+x, & 2, & 2+x, & 1+2x\}, \\
\{1+2x, & 2+x, & 2+2x, & 1\}, \\
\{2, & 2x, & 0, & 2+x\}, \\
\{2+x, & 0, & x, & 2+2x\}, \\
\{2+2x, & x, & 2x, & 2\}, \\
\{0, & 1+2x, & 1, & x\}, \\
\{x, & 1, & 1+x, & 2x\}, \\
\{2x, & 1+x, & 1+2x, & 0\},
\end{array}
$$

(and from L_1)

$$
\begin{array}{llll}
\{1+2x, & x, & 2+x, & 2x\}, \\
\{1, & 2x, & 2+2x, & 0\}, \\
\{1+x, & 0, & 2, & x\}, \\
\{2+2x, & 1+x, & x, & 1+2x\}, \\
\{2, & 1+2x, & 2x, & 1\}, \\
\{2+x, & 1, & 0, & 1+x\}, \\
\{2x, & 2+x, & 1+x, & 2+2x\}, \\
\{0, & 2+2x, & 1+2x, & 2\}, \\
\{x, & 2, & 1, & 2+x\}.
\end{array}
$$

Using $f(a+bx) = ap + b = 3a + b$ as a labeling function we obtain the announcement in numbers as follows (note that numbers have been arranged in increasing order):

$$
\begin{gathered}
\{3,4,6,8\}, \{4,5,6,7\}, \{3,5,7,8\}, \\
\{0,2,6,7\}, \{0,1,7,8\}, \{1,2,6,8\}, \\
\{0,1,3,5\}, \{1,2,3,4\}, \{0,2,4,5\}, \\
\{1,2,5,7\}, \{0,2,3,8\}, \{0,1,4,6\}, \\
\{1,4,5,8\}, \{2,3,5,6\}, \{0,3,4,7\}, \\
\{2,4,7,8\}, \{0,5,6,8\}, \{1,3,6,7\}.
\end{gathered}
$$

Table 3.3 illustrates how this announcement satisfies CA4 for the (4,4,1) instance.

3.5 Other Announcements

Other announcements relevant to this dissertation were found in the literature. Most of them, however, were not designed with the combinatorial axioms in mind, so their

$$\textit{Announcement} : \{3468, 4567, 3578, 0267, 0178, 1268, 0135, 1234, 0245,$$
$$1257, 0238, 0146, 1458, 2356, 0347, 2478, 0568, 1367\}$$

		Occurrences									
X	D_X	0	1	2	3	4	5	6	7	8	n_X
0	{3468, 4567, 3578, 1268, 1234, 1257, 1458, 2356, 2478, 1367}		5	5	5	5	5	5	5	5	5
1	{3468, 4567, 3578, 0267, 0245, 0238, 2356, 0347, 2478, 0568}	5		5	5	5	5	5	5	5	5
2	{3468, 4567, 3578, 0178, 0135, 0146, 1458, 0347, 0568, 1367}	5	5		5	5	5	5	5	5	5
3	{4567, 0267, 0178, 1268, 0245, 1257, 0146, 1458, 2478, 0568}	5	5	5		5	5	5	5	5	5
4	{3578, 0267, 0178, 1268, 0135, 1257, 0238, 2356, 0568, 1367}	5	5	5	5		5	5	5	5	5
5	{3468, 0267, 0178, 1268, 1234, 0238, 0146, 0347, 2478, 1367}	5	5	5	5	5		5	5	5	5
6	{3578, 0178, 0135, 1234, 0245, 1257, 0238, 1458, 0347, 2478}	5	5	5	5	5	5		5	5	5
7	{3468, 1268, 0135, 1234, 0245, 0238, 0146, 1458, 2356, 0568}	5	5	5	5	5	5	5		5	5
8	{4567, 0267, 0135, 1234, 0245, 1257, 0146, 2356, 0347, 1367}	5	5	5	5	5	5	5	5		5

Table 3.3: Illustration of CA4 for the finite field design with $p = 3$ for $(4, 4, 1)$.

constructions were not examined in detail. They are included here so as to provide more experimental evidence in order to uncover patterns that could be useful in further research.

3.5.1 Designs by Yates

Five 2-designs were taken from an article by Yates (1936). They satisfy all five combinatorial axioms and thus agree with the theorems linking CA4 and 2-designs in the following chapter. It should be noted here that the 7-line announcement for $(3, 3, 1)$ and the 13-line announcement for $(4, 8, 1)$ seem to be isomorphic to the projective plane designs mentioned earlier. Thus, only three of the announcements were unknown before.

3.5.2 Designs by Bose

Two designs were found in an article by Bose (1939), one 2-design and one 3-design. Both satisfy all five combinatorial axioms. The 3-design is a 14-line announcement for $(4, 3, 1)$ and seems to be isomorphic to the corresponding binary design. The 2-design provides us with a solution for the $(7, 7, 1)$ instance which was previously not covered.

3.5.3 Extended Designs

Several of the designs that satisfy all five combinatorial axioms can be extended to produce new announcements for different instances by using the construction suggested by Alltop (1975). The 7-line projective plane for $(3, 3, 1)$ has been extended giving a 14-line announcement for $(4, 3, 1)$ which is probably isomorphic to the binary design for that case. Similarly, Bose's $(7, 7, 1)$ announcement extends to give a solution for $(8, 7, 1)$ for which we already have a binary design. The finite field solution for $(4, 4, 1)$ has been extended to give a new announcement for $(5, 4, 1)$ which satisfies all axioms except CA1. It should also be noted here that the number of lines in this announcement is equal to the upper bound for an announcement satisfying CA1, CA2, and CA3 as proposed in (Albert *et al.*, 2003). The other three finite field announcements in Table C.1 have also been extended and they satisfy CA2 through CA5. Yates' $(5, 5, 1)$ announcement has been extended giving a previously unknown solution for $(6, 5, 1)$.

3.5.4 Announcements for $(3, b, 1)$

Albert *et al.* proposed a method for constructing good announcements for these instances. Following their method five announcements have been constructed, none of which satisfy CA4 or CA5. Interestingly, some do not even satisfy CA1 as claimed in the construction. Also, none of them are 2-designs which reinforces the validity of the upcoming theorems linking CA4 and 2-designs for $c = 1$.

Chapter 4

Theoretical Results

Theorem 1. *CA4 holds if and only if CA5 holds.*

Proof. CA5 implies CA4. Assume CA5 holds. Let X be any set of c points. Let the number of sets in E_X that contain a particular card y be denoted by m_X. Note that for every line L in D_X there is a set $\Omega - X - L$ in E_X, and all sets in E_X have this form. That is, $|E_X| = |D_X|$. Also note that by construction the lines in D_X and sets in E_X have empty intersection with X. The line L and set $\Omega - X - L$ are complements over the set $\Omega - X$, i.e. if a card $y \notin X$ is not in L then it must be in $\Omega - X - L$ and vice versa. We now pick an arbitrary card $y \in \Omega - X$ which (by CA5) occurs in m_X sets in E_X. That means, that y does not occur in $|D_X| - m_X$ sets in E_X. Because of the complementary property, this means that y occurs in $|D_X| - m_X$ lines in D_X. And since both $|D_X|$ and m_X are independent of y, we now set $n_X = |D_X| - m_X$ and have shown that the number of lines in D_X that contain card y is also independent of y and therefore CA4 must hold as well.
CA4 implies CA5. The proof follows a similar argument as above. □

Theorem 2. *For all instances where $c = 1$, if CA4 holds and $\mathcal{L}$ is a 1-design, then $\mathcal{L}$ is a 2-design.*

Proof. Assume $\mathcal{L}$ is a 1-design. Then $r_y = |\{L \in \mathcal{L} \mid y \in L\}|$ is independent of y. Also assume CA4 holds so that every D_X is a 1-design. Note that the lines in D_X are given by $\{L \in \mathcal{L} \mid x \notin L\}$ and that the size of this set is independent of x, since $\mathcal{L}$ being a 1-design implies that the number of lines that do not contain card x is independent of x. The number of lines in D_X containing card y is $s_{xy} = |\{L \in \mathcal{L} \mid x \notin L, y \in L\}|$, which is independent of x and by CA4 independent of y. Now, let $t_{xy} = |\{L \in \mathcal{L} \mid x, y \in L\}|$ be the number of lines in $\mathcal{L}$ that contain both cards x and y. Note that, $\{L \in \mathcal{L} \mid y \in L\} = \{L \in \mathcal{L} \mid x, y \in L\} \cup \{L \in \mathcal{L} \mid x \notin L, y \in L\}$ and since the last two sets do not intersect we have $r_y = t_{xy} + s_{xy}$. Since r_y and s_{xy} are independent of x and y, so is t_{xy}. Thus, $\mathcal{L}$ is a 2-design. □

Since CA4 and CA5 imply each other by Theorem 1, we may arrive at a new theorem by simply replacing CA4 with CA5 in the above statement. The next theorem is a converse of Theorem 2.

Theorem 3. *For all instances where $c = 1$, if $\mathcal{L}$ is a 2-design then CA4 holds.*

Proof. Assume $\mathcal{L}$ is a 2-design, i.e. $|\{L \in \mathcal{L} \mid x, y \in L\}| = \lambda_2$ is independent of x and y. We want to show that $|\{L_D \in D_X \mid z \in L_D\}| = n_X$ is independent of z. Let $X = \{x\}$ be any holding of Cathy. Note that when she eliminates lines from $\mathcal{L}$ that contain x, she reduces the number of lines containing any $y \notin X$ by λ_2. Let $\lambda_1 = |\{L \in \mathcal{L} \mid y \in L\}|$, which is independent of y because $\mathcal{L}$ is also a 1-design. Before elimination λ_1 lines contained y. After elimination, $\lambda_1 - \lambda_2$ lines contain y. This is the number of lines n_X in D_X that contain y. Since it is independent of y, CA4 holds. $\square$

Note that CA4 implies CA5 by Theorem 1 and thus we can also draw conclusions about CA5 from the above theorem.

Theorem 4. *If CA4 holds then* $n_X = \frac{a|D_X|}{a+b}$.

Proof. Count the total number of cards occurring in D_X in two ways. Assuming CA4 holds there are $a + b$ distinct cards and each of them occurs n_X times. There are $|D_X|$ lines and each of them contains a cards. Thus $(a + b)n_X = a|D_X|$. $\square$

Theorem 5. *For all instances where $c = 1$, if CA4 holds then n_X is independent of X.*

Proof. Assume $c = 1$ and CA4 holds. Take two arbitrary distinct $X_1 = x_1$ and $X_2 = x_2$. Consider $D_{X_1} = \{L \in \mathcal{L} \mid x_1 \notin L\}$. It contains no lines that contain card x_1 and n_{X_1} lines that contain card x_2. It must therefore contain $|D_{X_1}| - n_{X_1}$ lines that contain neither card x_1 nor card x_2. And due to construction of the set, this is the exact number of lines in $\mathcal{L}$ that contain neither card. Now consider $D_{X_2} = \{L \in \mathcal{L} \mid x_2 \notin L\}$. It contains no lines that contain card x_2 and n_{X_2} lines that contain card x_1. It must therefore contain $|D_{X_2}| - n_{X_2}$ lines that contain neither card x_1 nor card x_2. And due to construction of the set, this is the exact number of lines in $\mathcal{L}$ that contain neither card. Thus, we get the following equation

$$\begin{aligned}
|D_{X_1}| - n_{X_1} &= |D_{X_2}| - n_{X_2} \\
n_{X_1}\frac{a+b}{a} - n_{X_1} &= n_{X_2}\frac{a+b}{a} - n_{X_2} \\
n_{X_1}\left(\frac{a+b}{a} - 1\right) &= n_{X_2}\left(\frac{a+b}{a} - 1\right) \\
n_{X_1}\left(\frac{b}{a}\right) &= n_{X_2}\left(\frac{b}{a}\right) \\
n_{X_1} &= n_{X_2}
\end{aligned}$$

Because x_1 and x_2 were chosen arbitrarily we conclude that n_X is independent of X. $\square$

Since n_X and $|D_X|$ are related by Theorem 4 we may conclude from the above that if CA4 holds, then $|D_X|$ is independent of X. And remembering that $|D_X| = |E_X|$ from the proof of Theorem 1 we may further deduce that if CA4 holds, then $|E_X|$ is also independent of X. The same holds when CA5 holds. Furthermore, combining this theorem with Theorem 3 we see that if $c = 1$ and $\mathcal{L}$ is a 2-design then n_X, $|D_X|$ and $|E_X|$ are independent of X.

Looking more closely at what Theorem 5 entails, we can see that if n_X is independent of X, then so is $|D_X|$. Note here, that because $c = 1$, $|D_X|$ is equal to the number of lines in $\mathcal{L}$ that do not contain the card in X. Thus, we conclude that the number of lines in $\mathcal{L}$ that do *not* contain a particular card y is independent of y. This implies that the number of lines in $\mathcal{L}$ that *do* contain a particular card y is also independent of y, therefore $\mathcal{L}$ is a 1-design. Now Theorem 2 applies and we see that $\mathcal{L}$ must be a 2-design. To summarize, if $c = 1$ and CA4 holds then $\mathcal{L}$ is a 2-design.

Theorem 6. *All binary designs are* 3*-designs.*

Proof. By construction binary designs are based on 2^n points and $2(2^n - 1)$ lines of size 2^{n-1}. We now show that any three points p_1, p_2, p_3 occur in exactly $2^{n-2} - 1$ lines. The lines of the design contain points of the form $x_1 x_2 \ldots x_n$, that is a point is represented by an n-vector $\mathbf{x}$. For any n-vector $\mathbf{b} \neq \mathbf{0}$ there are two lines:

$$\{\mathbf{x} \mid \mathbf{b} \cdot \mathbf{x} = 0\} \text{ which we call } z_{\mathbf{b}}$$

$$\{\mathbf{x} \mid \mathbf{b} \cdot \mathbf{x} = 1\} \text{ which we call } w_{\mathbf{b}}$$

Let the n-vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$ be three distinct points. A line containing these comes from a vector $\mathbf{b}$ with either

$$\mathbf{b} \cdot \mathbf{u} = 0 \qquad \mathbf{b} \cdot \mathbf{v} = 0 \qquad \mathbf{b} \cdot \mathbf{w} = 0 \qquad (z_{\mathbf{b}}) \qquad \text{or} \tag{4.1}$$

$$\mathbf{b} \cdot \mathbf{u} = 1 \qquad \mathbf{b} \cdot \mathbf{v} = 1 \qquad \mathbf{b} \cdot \mathbf{w} = 1 \qquad (w_{\mathbf{b}}) \tag{4.2}$$

We have two cases:

1. One of $\mathbf{u}, \mathbf{v}, \mathbf{w}$ is the sum of the other two.
   Then $\mathbf{u}, \mathbf{v}, \mathbf{w}$ form a subspace U of the vector space V of dimension n over the field of 2 elements. The subspace U has dimension 2. Note that the $z_{\mathbf{b}}$'s come from $U^{\perp} = \{\mathbf{b} \mid \mathbf{b} \cdot \mathbf{t} = 0 \text{ for all } \mathbf{t} \in U\}$ and that $\dim U + \dim U^{\perp} = n$. Therefore, the dimension of $U^{\perp}$ is $n - 2$ and the number of $z_{\mathbf{b}}$'s is $2^{n-2} - 1$ (excluding $\mathbf{b} = \mathbf{0}$). The number of $w_{\mathbf{b}}$'s is 0 because Equation 4.2 cannot hold.

2. The vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$ are linearly independent. Then $\mathbf{u}, \mathbf{v}, \mathbf{w}$ form a subspace of dimension 3 and by the same argument as above the number of $z_{\mathbf{b}}$'s is $2^{n-3} - 1$. The number of $w_{\mathbf{b}}$'s is 2^{n-3} because we just find one $\mathbf{b_0}$ for which Equation 4.2 holds and then add all the $z_{\mathbf{b}}$'s.

In both cases the number of lines that contain the three points is $2^{n-2} - 1$ in total. Thus, binary designs are 3-designs. □

Remember that 3-designs are also 2-designs. If we combine the above theorem with Theorem 3 we can conclude that a binary design will satisfy CA4 (and CA5) for $c = 1$.

Combinatorial Axiom 6. *Let* $D_X = \{L \in \mathcal{L} \mid L \cap X = \emptyset\}$ *as in CA4. For all* X, D_X *is a* 2*-design, i.e. the number of lines in* D_X *containing any pair* $\{a, b\}$ *where* $a, b \in \Omega - X$ *is independent of* a *and* b.

Note that by the properties of t-designs CA6 implies CA4 since if D_X is a 2-design it must also be a 1-design.

Theorem 7. *For all instances where $c = 1$, CA6 holds and $\mathcal{L}$ is a 1-design if and only if $\mathcal{L}$ is a 3-design.*

Proof. Assume $c = 1$ and $\mathcal{L}$ is a 3-design. Then the number of lines in $\mathcal{L}$ containing any triple $\{a, b, x\} \subseteq \Omega$ is independent of a, b and x. Let this number be denoted by $t_{abx} = |\{L \in \mathcal{L} \mid a, b, x \in L\}|$. Let $s_{abx} = |\{L \in \mathcal{L} \mid a, b \in L, x \notin L\}|$ be the number of lines in $\mathcal{L}$ containing a and b but not x. Note that this is the number of lines in D_X that contain a and b. The number of lines in $\mathcal{L}$ containing a and b is given by $r_{ab} = |\{L \in \mathcal{L} \mid a, b \in L\}| = s_{abx} + t_{abx}$. Because $\mathcal{L}$ is a 3-design, t_{abx} is independent of a, b and x. Note that $\mathcal{L}$ is also a 2-design, therefore r_{ab} is independent of a and b (and x). Thus, s_{abx} must be independent of a, b and x, i.e. D_X is a 2-design and CA6 holds.
Assume $c = 1$, CA6 holds and $\mathcal{L}$ is a 1-design. Then the number of lines in D_X containing a and b is independent of a, b and x and is given by $s_{abx} = |\{L \in \mathcal{L} \mid a, b \in L, x \notin L\}|$. Note that if D_X is a 2-design it is also a 1-design, therefore, CA4 holds as well. Since CA4 holds and $\mathcal{L}$ is a 1-design, it is also a 2-design. Thus, $r_{abx} = |\{L \in \mathcal{L} \mid a, b \in L\}|$ is independent of a, b and x. Let $t_{abx} = |\{L \in \mathcal{L} \mid a, b, x \in L\}|$. Since $r_{ab} = s_{abx} + t_{abx}$, t_{abx} must also be independent of a, b and x. Therefore, $\mathcal{L}$ is a 3-design. $\square$

From this and Theorem 6 we conclude that binary designs satisfy CA6 for $c = 1$.

Chapter 5

Conclusion and Further Work

This dissertation outlined the need for stricter requirements for cryptographic protocols inspired by the Russian cards problem. The previous requirements CA1, CA2, CA3, and the new requirements CA4 and CA5 were formally stated and informally explained. Three constructions that meet all the requirements were explained in detail. These are projective planes, finite field designs, and binary designs. Satisfaction of CA4 has been illustrated for an example of each category. Some other announcements were analyzed briefly. Several theorems relating to the new requirements and designs were proved.

CA4 and 2-Designs

All announcements found to satisfy CA4 so far have been 2-designs. For the $c = 1$ scenario it has been proved that if the announcement is a 2-design then CA4 is satisfied. It has also been shown that the converse is true. Further work might focus on whether these theorems hold for $c > 1$.

Binary Designs and 3-Designs

We have shown that all binary designs are 3-designs. They are also the only designs in Table C.1 that work for more than one instance of the (a, b, c) problem. It would be interesting to investigate whether there is a relationship between these two facts.

n_X's Independence of X

This dissertation has shown that, given CA4, n_X is independent of X when $c = 1$. Further research might investigate whether this holds for $c > 1$ as well.

Satisfaction of CA4 when $c > t$

It can be seen from Table C.1 that all announcements satisfying CA4 have $c \leq t$. Therefore, it is possible that CA4 cannot hold when $c > t$. Further research might look into this. Note that the converse, CA4 must hold when $c \leq t$, is not true, i.e. some announcements satisfy CA4 for $c \leq t$ and some do not.

References

Albert, M., Aldred, R., Atkinson, M., van Ditmarsch, H., and Handley, C. (2003). Safe communication for card players by combinatorial designs for two-step protocols. *Otago University Technical Reports*.

Alltop, W. O. (1975). Extending t-designs. *Journal of Combinatorial Theory A*, *18*, 177–186.

Assmus, Jr, E. F. and Key, J. D. (1992). *Designs and Their Codes*. New York, NY, USA: Cambridge University Press.

Bose, R. C. (1939). On the construction of balanced incomplete block designs. *Annals of Eugenics*, *9*, 353–399.

Hughes, D. (1962). t-designs and permutation groups. In *Proceedings of Symposia in Pure Mathematics*, 39–41. American Mathematical Society.

van Ditmarsch, H. (2003). The Russian cards problem. *Studia Logica*, *75*, 31–62.

Wallis, W. (1988). *Combinatorial Designs*. New York: M. Dekker.

Wikipedia (2005). Public-key cryptography. `http://en.wikipedia.org/wiki/Public-key_cryptography`. Accessed on 14. September 2005.

Yates, F. (1936). Incomplete randomized blocks. *Annals of Eugenics*, *7*, 121–140.

Appendix A

Notation

$\boldsymbol{a, b, c}$ are the number of cards held by Alice, Bob, and Cathy, respectively.

$\boldsymbol{\mathcal{L}}$ is the set of lines contained in the announcement.

$\boldsymbol{X}$ is the set of cards held by Cathy.

$\boldsymbol{D_X = \{L \in \mathcal{L} \mid L \cap X = \emptyset\}}$ is the set of all lines in the announcement that have zero intersection with X. That is, it contains those lines from the announcement that Cathy cannot exclude and that are thus possible hands of Alice.

$\boldsymbol{n_X}$ is the number of lines in D_X that contain a particular card y.

$\boldsymbol{\Omega}$ is the set of all cards.

$\boldsymbol{E_X = \{\Omega - L - X \mid L \in D_X\}}$ is the set of all lines produced by taking all cards except those that are in X and those that are in a particular line in D_X. That is, it holds possible hands of Bob.

Appendix B

Fields

Introduction

A field consists of a set of elements with addition (+) and multiplication ($*$) operations. Addition and multiplication are commutative and associative. There exist a zero element (0) and a one element (1). Every member x of the set has an additive inverse $(-x)$ so that $x + (-x) = 0$. Every member x of the set (except 0) has a multiplicative inverse (x^{-1}) such that $x * x^{-1} = 1$. Examples of fields are $\mathbb{R}$ (the set of real numbers) and $\mathbb{Q}$ (the set of rational numbers), both of which are infinite. There are also finite fields as discussed in more detail below.

Finite Fields

In finite fields not all numbers of the form $1+1+1+\dots$ are distinct. There exist m and n such that $\underbrace{1+1+1+\dots}_{m\ times} = \underbrace{1+1+1+\dots}_{n\ times}$ for $m \neq n$ and thus $0 = \underbrace{1+1+1+\dots}_{n-m}$ assuming $m < n$. The characteristic of the finite field is the smallest number $p > 0$ with $\underbrace{1+1+1+\dots}_{p} = 0$. An example of a finite field of size p is $\mathbb{Z}_p$ which consists of the numbers $0, 1, 2, \dots, p-1$ added and multiplied discarding multiples of p. Some facts about finite fields are:

- The characteristic p of a finite field is always a prime number.
- Every finite field has size p^k for $k \in \mathbb{Z}^+$.

Construction

The elements of a field are polynomials of degree less than k with coefficients from the prime field $\mathbb{Z}_p$. These elements can be added as usual by adding the polynomials but taking the coefficients modulo p. Multiplication is a little more tricky because we might end up with polynomials of degree greater than k. To avoid this, we choose a polynomial of degree k that cannot be factored and call it $f(x)$. Such polynomials

exist for all degrees. Then after multiplying two polynomials we remainder by dividing by $f(x)$ which will give us a polynomial with degree less than k again. There are p^k distinct polynomials because we have k coefficients and each coefficient can take values from 0 up to $p-1$. Thus, there are p^k elements in the field. The field $\mathbb{Z}_p$ itself is a field of size p with $k=1$. Its elements are polynomials of order 0, i.e. just the coefficients of which there are p, namely the numbers from 0 to $p-1$. Another important fact is that every field has an element $t(x)$ such that the powers of $t(x)$ give all nonzero elements in the field.

Appendix C

Announcements Tested

Table C.1 gives a detailed list of the designs tested, their parameters, and results for CA1–5. The names assigned refer to the type of the design or the origin as follows:

Proj. Plane (Projective Plane) Designs as described in Section 3.2. The actual announcements were obtained in personal communication with Mike Atkinson.

Finite Field, Binary These designs refer to the constructions as they are described in Section 3.4 and Section 3.3, respectively. Note that binary designs have only been tested for instances with $b > c$ as this is a necessary requirement for the satisfaction of CA1 and CA2 (Albert *et al.*, 2003).

Yates Taken from *Incomplete randomized blocks* by Yates (1936).

Bose Taken from *On the construction of balanced incomplete block designs* by Bose (1939).

Ext. (Extended) Produced by extending existing designs as outlined in *Extending t-designs* by Alltop (1975).

$(3, b, 1)$ These follow the construction as described in (Albert *et al.*, 2003).

Remember that a t-design is also a 1-design, 2-design, ..., $(t-1)$-design. The value of t noted here is the maximum and the value for λ is the one pertaining to that maximum t-design.
Satisfaction of the combinatorial axioms (CAs) and t-design properties have been tested with the help of several computer programs. The C++ source code can be requested from the author. Note that due to the large number of combinations that have to be generated in some instances not all announcements have been tested completely. Two programs have been written to generate binary designs with given parameter n and finite field designs with given parameter p. They implement the constructions explained in Sections 3.3.2 and 3.4.2. The announcements themselves as well as the construction programs are also available for further research.

Design	(a,b,c) Instance	# of lines	t	λ	CA1	CA2	CA3	CA4	CA5
Proj. Plane	(3,3,1)	7	2	1	YES	YES	YES	YES	YES
Proj. Plane	(4,8,1)	13	2	1	YES	YES	YES	YES	YES
Finite Field	(4,4,1)	18	2	3	YES	YES	YES	YES	YES
Finite Field	(12,12,1)	50	2	11		YES	YES	YES	YES
Finite Field	(24,24,1)	98	2	23		YES	YES	YES	YES
Finite Field	(60,60,1)	242	2	59		YES	YES	YES	YES
Binary	(4,3,1)	14	3	1	YES	YES	YES	YES	YES
Binary	(8,7,1)	30	3	3	YES	YES	YES	YES	YES
Binary	(8,6,2)	30	3	3	YES	YES	YES	YES	YES
Binary	(8,5,3)	30	3	3	YES	YES	NO	NO	NO
Binary	(16,15,1)	62	3	7		YES	YES	YES	YES
Binary	(16,14,2)	62	3	7		YES	YES	YES	YES
Binary	(16,13,3)	62	3	7		YES	YES	YES	YES
Binary	(16,12,4)	62	3	7		YES	NO	NO	NO
Binary	(16,11,5)	62	3	7		NO	NO	NO	NO
Binary	(16,10,6)	62	3	7		NO	NO	NO	NO
Binary	(16,9,7)	62	3	7		NO	NO	NO	NO
Binary	(32,31,1)	126	3	15		YES	YES	YES	YES
Binary	(32,30,2)	126	3	15		YES	YES	YES	YES
Binary	(32,29,3)	126	3	15		YES	NO	NO	NO
Binary	(32,28,4)	126	3	15		YES	NO	NO	NO
Binary	(32,27,5)	126	3	15		YES	NO	NO	NO
Yates	(3,3,1)	7	2	1	YES	YES	YES	YES	YES
Yates	(4,8,1)	13	2	1	YES	YES	YES	YES	YES
Yates	(5,5,1)	11	2	2	YES	YES	YES	YES	YES
Yates	(5,15,1)	21	2	1	YES	YES	YES	YES	YES
Yates	(6,9,1)	16	2	2	YES	YES	YES	YES	YES
Bose	(4,3,1)	14	3	1	YES	YES	YES	YES	YES
Bose	(7,7,1)	15	2	3	YES	YES	YES	YES	YES
Ext. (3,3,1)	(4,3,1)	14	3	1	YES	YES	YES	YES	YES
Ext. (4,4,1)	(5,4,1)	36	3	3	NO	YES	YES	YES	YES
Ext. (5,5,1)	(6,5,1)	22	3	2	YES	YES	YES	YES	YES
Ext. (7,7,1)	(8,7,1)	30	3	3	YES	YES	YES	YES	YES
Ext. (12,12,1)	(13,12,1)	100	3	11		YES	YES	YES	YES
Ext. (24,24,1)	(25,24,1)	196	3	23		YES	YES	YES	YES
Ext. (60,60,1)	(61,60,1)	484	3	59		YES	YES	YES	YES
(3,b,1)	(3,3,1)	5	0		YES	YES	YES	NO	NO
(3,b,1)	(3,4,1)	6	0		NO	YES	YES	NO	NO
(3,b,1)	(3,5,1)	6	1	2	YES	YES	YES	NO	NO
(3,b,1)	(3,7,1)	8	0		NO	YES	YES	NO	NO
(3,b,1)	(3,8,1)	8	1	2	YES	YES	YES	NO	NO

Table C.1: Satisfaction of CA1–5 for all announcements tested.

The case of the hidden hand

Hans P. van Ditmarsch

Department of Computer Science
University of Otago
PO Box 56
Dunedin
New Zealand

hans@cs.otago.ac.nz

ABSTRACT. *In unconditionally secure protocols, a sender and receiver are able to communicate their secrets to each other without the eavesdropper(s) being able to learn the secret,* even *when the eavesdropper intercepts the entire communication. We investigate such protocols for the special case of deals of cards over players, where two players aim to communicate to each other their hand of cards without the remaining player(s) learning a single card from either hand. In this contribution we show that a particular protocol of length strictly larger than two (i.e., consisting of more than* just *one announcement by one player, and one other announcement by the other player) is after all not acceptable, and therefore does* not *constitute a new solution. The demonstration requires a detailed case-based analysis. The result may bring a general approach to arbitrary finite-length protocols closer.*

KEYWORDS: *cryptography, epistemic logic, epistemic action, secure protocols.*

1. Introduction

Assume we are given a group of agents. Will two of those agents be able to communicate a secret to each other? *Unconditionally secure protocols* guarantee that secrets are kept even under worst-case-scenario conditions where *all* communications are intercepted by an eavesdropper who also has computationally *unlimited* powers. Information-based methods to describe and verify protocols have been investigated in [BUR 90, RAM 01], and our work attempts to contribute to that tradition. We investigate particular instances of unconditionally secure protocols, namely where the agents are players, and the secrets correspond to aspects of card deals. The logical and combinatorial aspects of such protocols have been investigated in [FIS 96, DIT 03, MIZ 02, ALB 05, OTT 04], and for the automated verification of such protocols in epistemic model checkers we refer to [DIT 05b].

One particular setting concerns three players and seven cards. The analysis in dynamic epistemic logic in [DIT 03] shows that all announcements towards the exchange of secrets can, for each agent, be seen as (announcements of) a set of alternative local states. It also enumerates the protocols of length two for that case, and analyzes by logical means various unsatisfactory protocols. We think that these analyses are not trivial: the meaning of some sequence of announcements that constitutes a protocol may be rather unclear, because the interpretation of an announcement that is made towards a solution of the problem (i.e., towards the exchange of secrets), also depends on the commonly known intentions of rational agents executing a protocol resulting in such an announcement. These intentions, standardly delegated to the *pragmatics* of communication, can be drawn into the *semantics* of utterances in our dynamic epistemic analyses. In plain words: what you mean, is more than what you say.

There remain some open questions related to the dynamics of such card protocols that have not been answered in the cited publications. It remained unclear whether for the specific case investigated in [DIT 03] there were other solutions than those listed there, and even what 'other' means in this context. The postcondition that the secret has been exchanged need not be publicly known (i.e., including the eavesdropper), but it may be sufficient that it is only commonly known between sender and receiver. Can, subject to some notion of combinatorial equivalence, all protocols to communicate a given secret be enumerated? Can secrets always be exchanged by length-two protocols, or do some require strictly greater length? This contribution investigates the last question, for the specific case of three players and seven cards. We show that a particular protocol of length strictly larger than two, that was previously thought to be a promising candidate for a 'new solution', is after all not acceptable as a protocol. Note that the protocols in [FIS 96, MIZ 02] are also of greater length, but that no claim of minimality is made, and that (we confirmed that at least) some of their protocols can be reduced to length-two protocols.

A definite answer to this question is relevant for the systematic verification of such protocols when using model checkers, but also for the combinatorial design of protocols. In case it can be proved that all card protocols are essentially of length two, this would greatly constrain search when enumerating all protocols, for example by means of a model checker. In case it can be proved that longer protocols can be really different, scenarios are conceivable wherein a sender and receiver have achieved common knowledge of a secret, but where the eavesdropper is uncertain whether this has been achieved and still considers it possible that the protocol has not been finished – thus having to waste resources by keeping an eye on the communicating agents, waiting for possible further communications to intercept. As already said, we do not provide this definite answer, but hope that our contribution will bring one closer.

Section 2 introduces the three players and seven cards setting, and the problem that will be solved in this contribution: the protocol with the extra hand – figuratively called the 'hidden hand'. Section 3 defines public announcement logic and specifies protocol requirements in this logic. We then continue in Section 4 with the in-depth analysis of the specific protocol with one more hand in the initial announcement. Sec-

tion 5 presents a formula formalizing our result. It expresses the conditions for a secure announcement in a length-four protocol. We also suggest further lines of research for arbitrary finite protocols.

2. The case of the hidden hand

> *From a pack of seven known cards two players each draw three cards and a third player gets the remaining card. How can the players with three cards openly inform each other about their cards, without the third player learning from any of their cards who holds it?*

This 'Russian Cards' problem originated at the Moscow Math Olympiad 2000. Call the players Anne, Bill and Cath, and the cards $0, ..., 6$, and suppose Anne holds $\{0, 1, 2\}$, Bill $\{3, 4, 5\}$, and Cath card 6. For the hand of cards $\{0, 1, 2\}$, write 012 instead, for the card deal, write 012.345.6, etc. All announcements must be public and truthful. There are not many things Anne can safely say. Obviously, she cannot say "I have 0 or 6," because then Cath learns that Anne has 0. But Anne can also not say "I have 0 or 3," because Anne does not know if Cath has 3 or another card, and *if* Cath had card 3, she would have learnt that Anne has card 0. But Anne can also not say "I have 0 or 1." Even though Anne holds both 0 and 1, so that she does not appear to risk that Cath eliminates either card and thus gains knowledge about single card ownership (weaker knowledge, about alternatives, is allowed), Cath knows that Anne will not say anything from which Cath may learn her cards. And thus Cath can conclude that Anne will only say "I have 0 or 1" if she actually holds both 0 and 1. And in that way Cath learns two cards at once! The apparent contradiction between Cath not knowing and Cath knowing is not really there, because these observations are about different information states: it is merely the case that announcements may induce further updates that contain yet other information. There are various solutions that consist of first Anne and then Bill making an announcement, but – just to challenge the reader – all of the following are no solutions and run into trouble of the aforementioned kind (for details, see [DIT 03]): *Anne says that either she or Bill holds 012, after which Bill says that either he or Anne holds 345*, and also *Anne says that she does not hold card 6, after which Bill says that he does not hold card 6 either*, and also *Anne says that she either holds 012 or not any of those cards, after which Bill says that Cath holds card 6*. In all those cases, it turns out that, already after Anne's announcement, it is (at least) not common knowledge that Cath is ignorant of any of Anne's or Bill's cards, and that this is informative to Cath. Indeed, the solution requirement should be that Cath's ignorance remains common knowledge after any announcement. Such announcements are called *safe*. Further, one can prove that all informative announcements are equivalent to one of the form "my hand of cards is one of the following alternatives," so that all solutions consist of alternating statements of the players in that form. A combinatorial equivalent for a safe announcement consisting of hands, is (restricted to the set of cards that are not publicly known to be held by Cath): for each card, in the set of hands not containing that card, all other cards occur in at least one hand and are absent in at least one hand. A solution to the Russian Cards problem is a

sequence of safe announcements after which it is commonly known to Anne and Bill that Anne knows Bill's hand and Bill knows Anne's hand. The following is a solution:

> *Anne says "My hand of cards is one of 012, 034, 056, 135, 246," after which Bill says "Cath has card 6."*

Note that Bill's announcement is equivalent to "My hand of cards is one of 345, 125, 024." After Anne's announcement, Bill knows Anne's hand because one of his cards 3, 4, and 5 occurs in all hands except 012. After Bill's announcement, Anne knows Bill's hand as well, as 3, 4, and 5 are the remaining cards not held by Cath. After both Anne's and Bill's announcement, it is common knowledge that Cath does not know any of their cards. This can be proven by checking the combinatorial requirements. For example, after Anne's announcement, if Cath holds 0, the remaining hands are 135 and 246. Each of the cards $1, 2, ..., 6$ both occurs in at least one of 135 and 246 and is absent in at least one of those: 1 occurs in 135 and is absent in 246, 2 occurs in 246 and is absent in 135, etc. If Cath holds card 1, etc. After Bill's announcement this check is even easier. So both announcements are safe. Also, after both announcements it is common knowledge to Anne and Bill (and even public knowledge, i.e., common knowledge to Anne, Bill and Cath) that Anne knows Bill's hand, and vice versa.

If we *remove a single hand* from Anne's announcement, it can easily be seen that Cath will learn one or more of Anne's cards. For example, let us remove 246. Cath can now reason as follows: "Suppose Anne does not have 0. Then Anne can imagine that I have 0, in which case I could have eliminated all but 135 and learnt her cards. So if she does not have 0, she would never have said that. But she just did. So she must have 0. So I learnt one of her cards after all!" Similarly, Cath can now conclude that Anne holds 1, and so learns Anne's entire hand. If one removes another hand instead, a similar argument follows.

Now consider what happens if we *add a single hand* to Anne's announcement, a 'hidden hand' so to speak, as without it, her announcement already served its purpose; it therefore appears to be unrelated to the protocol underlying the previous announcement. For example, we add 245:

> *Anne says "My hand of cards is one of 012, 034, 056, 135,* **245**, *246," after which Bill says "Cath has card 6."*

As Anne's announcement is now slightly weaker, it is tempting to conclude that it is therefore less informative than her previous announcement. But is this really so?

First, let us assume that it is common knowledge to all players that after both announcements 'the problem will be solved', or, in other words, that the underlying protocol is of length two. On this assumption Cath actually learns some of Anne's cards: If Anne holds 245 or 246, then Anne can imagine (does not know not) that Bill has not learnt her hand, namely if Bill holds 013. Therefore, if the solution is known in advance to be of length two, Anne does not hold 245 or 246, but one of 012 034 056 135. Cath knows all of that too. But that is precisely the four hand announcement

just discussed. That was proven unsafe: Cath learnt Anne's entire hand of cards! So she will now, again. We see that instead of being *less* informative, Anne's six hand announcement is actually *more* informative than her five hand announcement. This is because Cath can assume that Anne's six hand announcement must have been informative enough for Bill *always* to learn her cards.

Next, suppose that we do not assume that the underlying protocol is of length two. Even though Anne knows that Bill knows her hand of cards, Cath can imagine (does not know not) that Anne does not know that: Cath, who holds 6, can imagine that Anne holds 245 and Bill 013, in which case Bill would not have learnt Anne's hand, so that a fortiori Cath can imagine that Anne can imagine that Bill has not learnt Anne's hand. Other choices of the sixth hand give slightly different results, but it always follows that it is not commonly known that Bill knows Anne's hand. On the other hand, we can compute in a way similar to that for the five hand protocol, that both after Anne's and after Bill's announcement it now remains common knowledge that Cath does not know any of Anne's or Bill's cards, and that after Bill's it is common knowledge that Anne knows Bill's hand and Bill knows Anne's. So, on the assumption that it is not commonly known that the protocol is of length two, we have found a solution of the Russian Cards problem of length two. This solution is different from the previous solution, because the intermediate information states are different: after Anne's announcement in the first sequence it is public knowledge that Bill knows Anne's cards, but after Anne's announcement in the second sequence this is *not* public knowledge (but only common knowledge for Anne and Bill). So we have found a new solution to the Russian Cards Problem!

Or haven't we?

The 'hidden hand' 245 – hidden because it appears not to be actually used in the protocol – only makes sense, if, when not 012.345.6 but instead, for example, 245.013.6 were the actual deal, there is a continuation of the communication between Anne and Bill, starting with Anne's six hand announcement, that also results in the solution requirements. Because if not, and because all three players are rational, then that hand 245 can be ruled out after all from public consideration, and both sequences would then be 'essentially' the same, i.e., describing identical information state transitions. This requires a systematic investigation of *all* possible continuations of that dialogue, which is exactly what we will undertake in this contribution. It turns out we only need to investigate protocols up to length four. But before we continue the exposition, we introduce the logic of public announcements in which this discussion finds a convenient and much more intelligible formal setting, so that we can do without the precise but sometimes confusing descriptions in natural language that we have used so far.

3. Public announcement logic

Given a set of *agents* N and a set of (propositional) *atoms* P, our basic structure is the *epistemic model* $M = \langle S, \sim, V \rangle$, where S is a *domain* of (factual) *states*, $\sim : N \to$

$\mathcal{P}(S \times S)$ defines a set of *accessibility* relations $\sim_n$ that are equivalence relations, and $V : P \rightarrow \mathcal{P}(S)$ defines a set of *valuations* $V_p \subseteq S$. A pointed structure (M, s) is called an *epistemic state*. The logical language is inductively defined as

$$\varphi \;::=\; p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid K_n\varphi \mid C_G\varphi \mid [\varphi]\psi$$

For $K_n\varphi$ read 'agent n knows φ', for $C_G\varphi$ (where G may be any subset of the 'public' N of all agents) read 'group G commonly know φ', and for $[\varphi]\psi$ read 'after (every) announcement of φ, it holds that ψ'. For the dual $\neg K_n\neg\varphi$ of 'knowing that', read 'agent n can imagine that φ', and we also write $\hat{K}_n\varphi$ for that. The semantics of this *multiagent logic of public announcements* is

$$\begin{array}{lll}
M, s \models p & : \text{iff} & s \in V_p \\
M, s \models \neg\varphi & : \text{iff} & M, s \not\models \varphi \\
M, s \models \varphi \wedge \psi & : \text{iff} & M, s \models \varphi \text{ and } M, s \models \psi \\
M, s \models K_n\varphi & : \text{iff} & \text{for all } t \sim_n s : M, t \models \varphi \\
M, s \models C_G\varphi & : \text{iff} & \text{for all } t \sim_G s : M, t \models \varphi \\
M, s \models [\varphi]\psi & : \text{iff} & M, s \models \varphi \text{ implies } M|\varphi, s \models \psi
\end{array}$$

where $\sim_G$ is the reflexive and transitive closure of the union of all $\sim_n$, i.e., $\sim_G := (\bigcup_{n \in G} \sim_n)^*$, and $M|\varphi$ is the restriction of M to the states where φ is true, i.e., $M|\varphi := \langle S', \sim', V' \rangle$ such that

$$\begin{array}{lll}
S' & := & \{v \in S \mid M, v \models \varphi\} \\
\sim'_n & := & \sim_n \cap (S' \times S') \\
V'_p & := & V_p \cap S'
\end{array}$$

From the various principles that hold for this logic, we merely mention two validities that we will refer to in the continuation: $[C_N\varphi]C_N\varphi$ says that the announcement of something that is already publicly known is not informative, and $[\varphi][\psi]\chi \leftrightarrow [\varphi \wedge [\varphi]\psi]\chi$ says that the postconditions of $[\varphi][\psi]$ and $[\varphi \wedge [\varphi]\psi]$ are the same, or in other words: if you first announce φ and after that ψ, you might as well have announced all at once $\varphi \wedge [\varphi]\psi$. Further note that $[\varphi]\varphi$ is *not* valid.[1] Public announcement logic is a special case of the action model logics presented in [BAL 99, BAL 04]. For an overview of the logic and a direct and succinct completeness proof, see [DIT 05a].

For the Russian Cards example, there are three agents and 21 atoms (seven cards times three agents). Atom q_n describes the fact that agent n holds card q, and $ijk_n := i_n \wedge j_n \wedge k_n$ describes that player n's hand is $\{i, j, k\}$, so that Anne holding card 0 is described by 0_a, and that Anne's hand is 012 is described by 012_a, etc. The structures on which we interpret such descriptions consist of a domain containing all deals of cards $Q = \{0, 1, 2, 3, 4, 5, 6\}$ over players $N = \{a, b, c\}$ (for Anne, Bill,

1. A well-known counterexample is the (dynamic reading of the) Moore-sentence: after an announcement of $p \wedge \neg K_a p$, this formula is no longer true: the agent now knows that p. For another example, the last of the introductory examples is *not* a solution because $Rus|K_a(012_a \vee \neg(0_a \vee 1_a \vee 2_a)), 012.345.6 \models [K_a\mathsf{cignorant}]\neg K_a\mathsf{cignorant}$.

and Cath, respectively). The equivalences on this domain are induced by players being able to see their own cards, and how many cards other players have. Therefore we can restrict ourselves to the (connected) model consisting of the $\binom{7}{3} \cdot \binom{4}{3} \cdot \binom{1}{1} = 140$ card deals where Anne and Bill each hold three and Cath one card. We call this model Rus (for 'Russian'). E.g., the a-equivalence class of deal 012.345.6 is $\{012.345.6, 012.346.5, 012.356.4, 012.456.3\}$, whereas the b-equivalence class of that deal is $\{012.345.6, 016.345.2, 026.345.1, 126.345.0\}$, and its c-equivalence class contains $\binom{6}{3} = 20$ card deals.

To give some examples, we can now describe that Anne knows that she holds card 0 by $K_a 0_a$, that Anne considers it possible that Bill holds card 6, even though he actually does not hold that card, by $\neg 6_b \wedge \hat{K}_a 6_b$, and that the players have common knowledge (background knowledge) that Anne knows her own hand of cards, as $C_{abc} \bigwedge_{i \neq j \neq k \in Q}(ijk_a \rightarrow K_a ijk_a)$.

The epistemic requirements for a problem solution are that Anne knows Bill's hand of cards, that Bill knows Anne's hand of cards, and that Cath is ignorant of any card held by Anne or Bill:

$$\begin{array}{lcl} \mathsf{aknowsbs} & := & \bigwedge_{i \neq j \neq k \in Q}(ijk_b \rightarrow K_a ijk_b) \\ \mathsf{bknowsas} & := & \bigwedge_{i \neq j \neq k \in Q}(ijk_a \rightarrow K_b ijk_a) \\ \mathsf{cignorant} & := & \bigwedge_{q \in Q} \bigwedge_{n = a,b} \neg K_c q_n \end{array}$$

When an agent n (a or b) is saying φ, this is interpreted as the announcement of $K_n \varphi \wedge [K_n \varphi] C_{abc} \mathsf{cignorant}$. Using the validities above, we see that

$$[K_n \varphi \wedge [K_n \varphi] C_{abc} \mathsf{cignorant}] \psi$$

is equivalent to

$$[K_n \varphi][C_{abc} \mathsf{cignorant}] \psi \ .$$

Using the validity of $[C_{abc}\mathsf{cignorant}] C_{abc}\mathsf{cignorant}$, this is equivalent to

$$[K_n \varphi](C_{abc} \mathsf{cignorant} \wedge \psi) \ ,$$

so that we can characterize a 'safe announcement' as one that is true and after which $\mathsf{cignorant}$ is common knowledge. A solution is a sequence of safe announcements after which $C_{ab}(\mathsf{aknowsbs} \wedge \mathsf{bknowsas})$ is true.[2]

We now can formalize the difference between the five hand and the six hand solution that we investigate. Define

$$\begin{array}{lcl} \mathsf{anne5} & := & K_a(012_a \vee 034_a \vee 056_a \vee 135_a \vee 246_a) \\ \mathsf{anne6} & := & K_a(012_a \vee 034_a \vee 056_a \vee 135_a \vee 245_a \vee 246_a) \\ \mathsf{bill} & := & K_b 6_c \end{array}$$

2. In [DIT 03] only $(\mathsf{aknowsbs} \wedge \mathsf{bknowsas})$ is required as a postcondition. The stronger postcondition $C_{ab}(\mathsf{aknowsbs} \wedge \mathsf{bknowsas})$ should also obviously hold (clearly, from the perspective of the communicating agents). It is unclear whether this makes a difference, and/or whether $C_{abc}(\mathsf{aknowsbs} \wedge \mathsf{bknowsas})$, with *public* knowledge, is even stronger.

These announcements are all safe. The two solution sequences can be abbreviated as anne5; bill and anne6; bill. Their difference appears from the models $Rus|\mathsf{anne5}$ and $Rus|\mathsf{anne6}$.

The epistemic model $Rus|\mathsf{anne5}$ can be schematically represented as

012.345.6	012.346.5	012.356.4	012.456.3			
034.125.6	034.126.5			034.156.2	034.256.1	
		056.123.4	056.124.3	056.134.2	056.234.1	
135.024.6		135.026.4		135.046.2		135.246.0
	246.013.5		246.015.3		246.035.1	246.135.0

Its domain consists of all states (card deals) where the announcement formula was true, i.e., where Anne holds one of the five hands in the announcement. The rows represent a-equivalence classes, all b-equivalence classes are singleton, and the columns represent c-equivalence classes. For example, it holds that $Rus|\mathsf{anne5}, 012.345.6 \models C_{abc}\mathsf{bknowsas}$, because all b-equivalence classes are singleton, so that whatever hand of cards Bill holds there is only one possible a-hand that he considers possible – therefore, he *knows* Anne to have that hand of cards.

The model $Rus|\mathsf{anne6}$ can be pictured as

012.345.6	012.346.5	012.356.4	012.456.3			
034.125.6	034.126.5			034.156.2	034.256.1	
		056.123.4	056.124.3	056.134.2	056.234.1	
135.024.6		135.026.4		135.046.2		135.246.0
245.013.6*			245.016.3		245.036.1	245.136.0
	246.013.5*		246.015.3		246.035.1	246.135.0

where all b-equivalence classes are singleton except $\{245.013.6, 246.013.5\}$. To indicate that, we have *-ed these two deals. We now have that $Rus|\mathsf{anne6}, 012.345.6 \models \hat{K}_c\neg K_a\mathsf{bknowsas}$, because $012.345.6 \sim_c 245.013.6$ (same column) and $Rus|\mathsf{anne6}$, $245.013.6 \models \neg K_a\mathsf{bknowsas}$, because $245.013.6 \sim_a 245.013.6$ and also $Rus|\mathsf{anne6}$, $245.013.6 \models \neg\mathsf{bknowsas}$. The last is true, because 245_a holds but not K_b245_a (because $245.013.6 \sim_b 246.013.5$ and $Rus|\mathsf{anne6}, 246.013.5 \not\models 245_a$), so that we have $Rus|\mathsf{anne6}, 245.013.6 \not\models 245_a \rightarrow K_b245_a$.

Both after $(\mathsf{anne5}; \mathsf{bill})$ and after $(\mathsf{anne6}; \mathsf{bill})$ the model is

012.345.6
034.125.6
135.024.6

in which all solution requirements are common knowledge (so, in particular, it is not just common knowledge to Anne and Bill that they know each other's hand of cards, but this is even publicly known to all players). This will be enough backbone to strengthen our exposition in Section 5, after first, in the next section, exhaustively exploring all further developments of protocols starting with anne6.

4. Uncovering the hidden hand

So, once more, as in Section 2, suppose that Anne says: "My hand is one of 012, 034, 056, 135, 245, 246," but that the actual deal is 245.013.6 instead of 012.345.6. In this scenario, Bill has not learnt that Cath's card is 6. How can Bill safely respond to Anne, and Anne to Bill, and so on? We will now systematically investigate all responses.

I do not know Cath's card

Bill cannot admit that he doesn't know Cath's card, or, equivalently, that he doesn't know Anne's hand, because he would then be giving away that Anne's hand must be either 245 or 246. Cath would therefore learn that Anne has cards 2 and 4. Lost.

Please say something again, Anne

He also cannot say *nothing*, or in more polite phrasing: "Please say something again, Anne." This is because Anne cannot respond to *that*: After Anne's announcement, Bill does not know whether Anne has 245 or 246. If Anne actually held 246 she could respond to Bill's request by saying, after all, that her hand is one of 012, 034, 056, 135, 246; i.e. she simply leaves out 245. But she holds 245, and no strict subset of $\{012, 034, 056, 135, 245, 246\}$ containing 245 is safe! This is easy to observe: Bill's response has only heuristic and no informative content. Therefore, we can analyze Anne's second announcement as if it were made in the original epistemic state $(Rus, 245.013.6)$. Suppose 012 were left out, then if Cath held 5 she would learn that Anne held 4; suppose 034 were left out, then if Cath held 2, she would learn that Anne held 5. Without 056, if Cath holds 2 she learns Anne holds 3, without 135, if Cath holds 0 she learns Anne holds 2. But $\{012, 034, 056, 135, 245\}$ is also unsafe; e.g., if Cath had 0, she would learn that Anne has 5. So she can't say anything, apart from "Please say something again, Bill." But then we are back where we started: what should Bill say?

Two obvious replies of Bill to Anne have been outruled now. Bill cannot say that he does not know Cath's card yet, but he can also not try to hide that information by being non-committal. It seems we have run out of options. But this is far from the case: there are *many* others! After Anne says that she has one of 012, 034, 056, 135, 245, and 246, the domain consists of 24 card deals. These make up 23 b-equivalence classes: all are singleton, except $\{245.013.6, 246.013.5\}$. An announcement of Bill is a set of alternative b-hands, i.e., is interpreted as a union of b-classes, and as Bill is truthful, the b-class that contains the actual deal must always be included. But that means that any subset of these 23 classes that includes $\{245.013.6, 246.013.5\}$ denotes a possible reply of Bill to Anne. That makes 2^{22} replies to choose from. So far, we have ruled out two: the subset $\{245.013.6, 246.013.5\}$ corresponding to "I don't know your hand, Anne," and the subset of *all* b-classes, corresponding to

"Please say something again, Anne." Note that we could rule out these replies by investigating any possible reply of Anne to that reply of Bill, and so on. There are therefore $2^{22} - 2$ remaining possible replies of Bill to investigate, and for each of those we have to consider all replies Anne can make to Bill's, and subsequent replies of Bill to Anne's second announcement, etc., ad infinitum. This appears rather intractable, if not undecidable at that...

Fortunately, we can systematically investigate all cases by further logical and combinatorial analysis [DIT 03]. One important observation is, that we can assume a maximum number of announcements in a protocol. (So that there is a decision procedure to determine the adequacy of arbitrary protocols.) This is because all informative announcements restrict the domain, because the domain is finite, and because uninformative replies of the kind "please say something again" (as above) are meaningless if they occur at least twice in a row. Another important observation is that announcements must be safe: most of the b-classes are deals; for a given card (that Cath may hold), we must have enough deals to ensure safety, i.e., to ensure Cath's ignorance if she held that card. Now a set of two deals is only safe if Anne's and Bill's hands are disjoint, i.e., if it has the form $\{ijk.lmn.o, lmn.ijk.o\}$. From the 24 deals that we consider here only $\{135.246.0, 246.135.0\}$ have that property (and the exception does not include the actual deal). So – keeping in mind the exception – if Bill's announcement includes one deal for some card q, it must contain at least two more deals where Cath holds q. (So, as well, exhaustive search for protocols may be more tractable than it appears on first sight.)

Now consider the following reply – it turns out this one will be safe, and it is typical for *all* other safe replies:

Cath has 5 or 6

Suppose Bill says "Cath has card 5 or card 6." This is true, and this is safe. Common knowledge of Cath's ignorance was already established after Anne's first announcement, and will obviously remain true for any union of b-classes that is a union of c-classes. The models underlying the epistemic states before and after the response are

012.345.6	012.346.5	012.356.4	012.456.3			
034.125.6	034.126.5			034.156.2	034.256.1	
		056.123.4	056.124.3	056.134.2	056.234.1	
135.024.6		135.026.4		135.046.2		135.246.0
245.013.6*			245.016.3		245.036.1	245.136.0
	246.013.5*		246.015.3		246.035.1	246.135.0

$\Downarrow \quad K_b(5_c \vee 6_c)$

```
012.345.6 012.346.5
034.125.6 034.126.5
135.024.6
245.013.6*
          246.013.5*
```

What can Anne say that is still safe? She cannot say: “Cath has card 6,” because that would make it public that she knows that, which would eliminate the a-classes $\{012.345.6, 012.346.5\}$ and $\{034.125.6, 034.126.5\}$ where she does not know that, so the resulting model would be

```
135.024.6
245.013.6
```

so that Cath knows that Anne holds card 5 and is no longer ignorant. Formally, we have that

$$Rus, 245.013.6 \models [\mathsf{anne6}][K_b(5_c \vee 6_c)][K_a 6_c]\neg\mathsf{cignorant}$$

because

$$Rus|\mathsf{anne6}|K_b(5_c \vee 6_c)|K_a 6_c, 245.013.6 \models \neg\mathsf{cignorant}$$

because $K_c 5_a$ holds in the last epistemic state. Is there anything else she can say in response? Yes: Anne can respond with “I do not have 135.” We then get the model

```
012.345.6 012.346.5
034.125.6 034.126.5
245.013.6*
          246.013.5*
```

This announcement is safe, because it remains common knowledge that Cath is ignorant. What can Bill say in return, after that? Unfortunately, nothing informative: any restriction of either of the current c-classes makes them unsafe – for the given card deals, any c-class with fewer than three elements is unsafe. Right, so we are down to $2^{22} - 3$ remaining cases. What next?

Variations on ‘Cath has 5 or 6’

What deal can we remove from Bill’s reply “Cath has 5 or 6” such that it remains safe? The actual b-class $\{245.013.6, 246.013.5\}$ must always be included (or Bill would be lying). Therefore, at least two deals where Cath holds 5 and at least two deals where Cath holds 6 have to be included as well, such that it remains common knowledge that she is ignorant (see above). There are only two other deals where Cath holds 5, that therefore both need to be included. There are three other deals where Cath holds 6, and any two out of these three keep Cath ignorant. For example, if Bill omits 135.024.6 (“My hand of cards is one of 345 346 125 126 024 013.”) we get

```
012.345.6 012.346.5
034.125.6 034.126.5
245.013.6*
          246.013.5*
```

and Anne cannot respond informatively to Bill at all, nor can Bill if Anne were to ask Bill to respond once more. Any strict subset of a-classes is unsafe, as before. If Bill omits 012.345.6 instead, we get

```
          012.346.5
034.125.6 034.126.5
135.024.6
245.013.6*
          246.013.5*
```

and again Anne cannot respond. The case where Bill omits 034.125.6 instead, is similar. Down to $2^{22} - 6$ ...

Other replies

We have now analyzed six possible replies, two of those were unsafe, four were safe but were leading nowhere. Now take any of those four, and consider adding any number of the remaining b-classes, i.e. any card deal where Cath does not hold 5 or 6. For example, taking the last epistemic model, we get, schematically

```
          012.346.5  +++
034.125.6 034.126.5  +++
                     ++
135.024.6            +++
245.013.6*           +++
          246.013.5* +++
```

where +++ and ++ denote possible other card deals included by Bill, and ++ specifically those where Anne holds 056. Very similar to the scenarios before, Anne cannot now say anything: the hands she will announce always must include her actual hand 245 and therefore as well 034 and 135, because any subset is unsafe, but that means she will have to carry along at least one deal where Cath holds 5 as well, namely 034.126.5. But therefore she will have to include enough other deals where Cath holds 5 so that the c-equivalence class for card 5 is safe, so therefore she must always include her hands 012 and 246 anyway. The only remaining hand is now 056. Suppose Bill had included, for example, 056.123.4 in his announcement, then he would have been obliged to include 012.356.4 and 135.026.4 as well, and because Anne's announcement includes hand 012 and 135 it therefore must include 056 as well. Suppose, instead, Bill had included 056.124.3 in his announcement, then ..., etc. So Anne

cannot delete *a single* from her six hands, i.e., she cannot make an informative announcement. The other cases are just as similar to another one of the 'Cath has 5 or 6' variations that we have already discussed. Either Anne cannot respond at all, as here, or Anne may be able to reply to that she does not have 135, to which then Bill cannot respond. That was, after all, rather quick for $2^{22} - 6$ remaining cases!

We have now established the following. If the card deal is 245.013.6, then after Anne has said "I have one of 012, 034, 056, 135, 245, 246," in whatever way Bill responds to that, either Anne cannot respond informatively, or Anne can make an informative response to which Bill then cannot respond. Therefore, no *effective* protocol for card deal 245.013.6 starts with Anne saying that she has one of 012 034 056 135 245 246. We assume that Anne and Bill take no risks: they are only willing to execute protocols that guarantee success, in the sense that, whatever one says, the other can make at least one safe reply to that which will bring a solution closer. Therefore Anne will not execute this protocol for any card deal wherein she holds 245. Therefore, hand 245 is publicly known *not* to be Anne's actual hand. That suggests that the models $Rus|\mathsf{anne6}$ and $Rus|\mathsf{anne5}$ are identical 'if we incorporate all information'. Given that $Rus|\mathsf{anne5}|\mathsf{bill}$ and $Rus|\mathsf{anne6}|\mathsf{bill}$ were the same anyway (namely the epistemic model with domain $\{012.345.6, 034.125.6, 135.024.6\}$), it would mean that $\mathsf{anne5}; \mathsf{bill}$ and $\mathsf{anne6}; \mathsf{bill}$ are 'essentially the same' in the sense that both induce the same epistemic state transitions. Thus we have uncovered the hidden hand.

But it turns out that things are, again, not as they seem. Because if Anne held hand 246 instead of 245, she would *as well* consider it possible that Bill holds 013. And when 246.013.5 is the actual deal of cards, an almost identical line of argument to the previous one reveals that again no exit of a sequence of safe announcements is *guaranteed*. Therefore, also hand 246 is publicly known not to be Anne's hand. But the remaining set of four hands 012 034 056 135 is unsafe, as was already explained in Sectionhand. Therefore Anne will not say *at all* that her hand is one of the six 012 034 056 135 245 246. By uncovering the hidden hand we have laid bare the foundations of the entire announcement, and see it collapse.

We conclude that $(\mathsf{anne6}; \mathsf{bill})$ does not solve the Russian Cards problem. This is a minor result, but a result all the same, and we have answered one of the remaining riddles concerning Russian Cards.

Suppose we *had* shown that both $\mathsf{anne5}$ and $\mathsf{anne6}$ are suitable first announcements of secure protocols. Why not excuse ourselves from using the second by the simple observation that it contains the first? A way that makes clearer why this would have been a result, is to turn matters around: Suppose we *had* established that $\mathsf{anne6}$ starts a sequence of announcements that provides a solution for deal 245.013.6. Then Anne could have executed the same underlying protocol if her hand had been 012 (as in deal 012.345.6) but then 'with 012 in the role of 245' in $\mathsf{anne6}$. That would have resulted in, for example, Anne saying: "My hand is one of 012, 026, 034, 135, 146, 245." None of the (more than one hundred) solutions listed in [DIT 03] start with Anne announcing a subset of that. So, indeed, that would have been a new solution to the problem.

5. Further research

The previous section ended on a suspicious note with phrasings like 'incorporate all information'. What does that really mean? Is that phrasing not just a clever trick to be excused from formal precision? We observed in Section 3 that an announcement of φ in the card protocols setting should be interpreted as $K_n\varphi \wedge [K_n\varphi]C_{abc}\mathsf{cignorant}$. Given that interpretation, the model $Rus|\mathsf{anne5}$ is contained in the model $Rus|\mathsf{anne6}$. Apparently, we need to incorporate even more pragmatic information into the meaning of φ, or in other words, φ should be interpreted as $K_n\varphi \wedge [K_n\varphi](C_{abc}\mathsf{cignorant} \wedge \psi)$, for some formula ψ. Subject to *that* interpretation, $Rus|\mathsf{anne6}$ is no longer contained in $Rus|\mathsf{anne5}$. Indeed, no model $Rus|\mathsf{anne6}$ even *exists* in that case, as the announcement cannot be truthfully (safely) uttered. What is ψ?

We constructed such a ψ above. Let ψ_1 represent an arbitrary informative response of Bill to Anne's announcement $\mathsf{anne6}$. In other words, given some card deal $ijk.lmn.o$, these are disjunctions containing a part lmn_b and various other parts pqr_b. For most card deals in $Rus|\mathsf{anne6}$ this includes a safe response of Bill to Anne such that (at least) the solution criterium $\mathsf{bknowsas} \wedge \mathsf{aknowsbs}$ holds afterwards.[3] For deal $245.013.6$ this does *not* include such a response. In that case, we get the descriptions of subsets of the 24 possible b-hands that include actual b-class $\{245.013.6, 246.013.5\}$, as exhaustively described in the previous section. And those formulas now always contain a part 013_b.

Further, let ψ_2 be an arbitrary informative reponse of Anne to Bill's response, and ψ_3 an arbitrary informative reponse of Bill to that. Note that for arbitrary φ formula $\varphi \wedge [\varphi]\psi$ is equivalent to $\langle\varphi\rangle\psi$. Then we have shown that the following formula is *false* in the model Rus, for $K_a\varphi = \mathsf{anne6}$. The formula expresses that a solution can be found by *some* protocol of depth at most four. We omit details.

$$\begin{array}{l}
\langle\hat{K}_a\varphi\rangle(C_{abc}\mathsf{cignorant} \wedge C_{abc}((\mathsf{bknowsas} \wedge \mathsf{aknowsbs})\vee \\
\quad \bigvee_{\psi_1}\langle\hat{K}_b\psi_1\rangle(C_{abc}\mathsf{cignorant} \wedge C_{abc}((\mathsf{bknowsas} \wedge \mathsf{aknowsbs})\vee \\
\quad\quad \bigvee_{\psi_2}\langle\hat{K}_a\psi_2\rangle(C_{abc}\mathsf{cignorant} \wedge C_{abc}((\mathsf{bknowsas} \wedge \mathsf{aknowsbs})\vee \\
\quad\quad\quad \bigvee_{\psi_3}\langle\hat{K}_b\psi_3\rangle C_{abc}\mathsf{cignorant} \quad))))))
\end{array}$$

In other words, Anne only says something, if it is safe and – unless the problem is solved – if Bill has at least one safe response to that, to which – unless the problem is now solved – Anne will be able to respond safely, and so on, until the problem is solved. In our example, 'depth four' sufficed to uncover a non-solution. In general, the depth required is finite and it is a function of the card deal. We are still uncertain about the general formula; these dynamic settings are rather sensitive to the presence of common and public knowledge requirements (such as, should it always be C_{abc}, or just C_{ab}). A more elegant formulation is to be expected in a more expressive (but

3. In fact for *all* card deals except $245.013.6$, $246.013.5$, $012.356.4$, and $034.126.5$. Please ignore the common knowledge closure condition C_{ab} – this makes no difference here.

undecidable) logic, namely the logic of public announcements with arbitrary iteration of announcements [MIL 05].

We hope to continue our investigations in these directions.

Acknowledgements

For the completion of this research I acknowledge support from research grant AOARD-05-4017. Sieuwert van Otterloo suggested the six hand example and the complication that anne6 ; bill appears to be a solution of length two on the assumption that a solution may be of greater length. I thank the anonymous referees supplied by the journal for their constructive and helpful comments. This article is largely based on a similarly titled contribution to the (informal) Liber Amicorum Dick de Jongh [DIT 04]; note that in that version we *mistakenly* claimed that anne6 was safe and could be reduced to anne5 – even though the analysis prior to that conclusion is identical to the one presented here.

6. References

[ALB 05] ALBERT M., ALDRED R., ATKINSON M., VAN DITMARSCH H., HANDLEY C., "Safe communication for card players by combinatorial designs for two-step protocols", *Australasian Journal of Combinatorics*, vol. 33, 2005, p. 33–46.

[BAL 99] BALTAG A., MOSS L., SOLECKI S., "The logic of public announcements, common knowledge, and private suspicions", report , 1999, Centrum voor Wiskunde en Informatica, Amsterdam, CWI Report SEN-R9922.

[BAL 04] BALTAG A., MOSS L., "Logics for epistemic programs", *Synthese*, vol. 139, 2004, p. 165–224, Knowledge, Rationality & Action 1–60.

[BUR 90] BURROWS M., ABADI M., NEEDHAM R., "A logic of authentication", *ACM Transactions on Computer Systems*, vol. 8, 1990, p. 18–36.

[DIT 03] VAN DITMARSCH H., "The Russian cards problem", *Studia Logica*, vol. 75, 2003, p. 31–62.

[DIT 04] VAN DITMARSCH H., "The Case of the Hidden Hand", *Liber Amicorum Dick de Jongh*, 2004, http://www.illc.uva.nl/D65/.

[DIT 05a] VAN DITMARSCH H., VAN DER HOEK W., KOOI B., "Dynamic Epistemic Logic", Manuscript, 2005.

[DIT 05b] VAN DITMARSCH H., VAN DER HOEK W., VAN DER MEYDEN R., RUAN J., "Model Checking Russian Cards", Presented at MoChArt 05 (Model Checking in Artificial Intelligence) and to appear in Electronic Notes in Theoretical Computer Science, 2005.

[FIS 96] FISCHER M., WRIGHT R., "Bounds on Secret Key Exchange Using a Random Deal of Cards", *Journal of Cryptology*, vol. 9(2), 1996, p. 71–99.

[MIL 05] MILLER J., MOSS L., "The Undecidability of Iterated Modal Relativization", *Studia Logica*, vol. 79(3), 2005, p. 373–407.

[MIZ 02] MIZUKI T., SHIZUYA H., NISHIZEKI T., "A complete characterization of a family of key exchange protocols", *International Journal of Information Security*, vol. 1, 2002, p. 131–142.

[OTT 04] VAN OTTERLOO S., VAN DER HOEK W., WOOLDRIDGE M., "Model Checking a Knowledge Exchange Scenario", *Applied Artificial Intelligence*, vol. 18(9-10), 2004, p. 937–952.

[RAM 01] RAMANUJAM R., SURESH S. P., "Information based reasoning about security protocols", *Electr. Notes Theor. Comput. Sci.*, vol. 55(1), 2001.

Model Checking Sum and Product

H.P. van Ditmarsch[1], J. Ruan[1,⋆], and L.C. Verbrugge[2,⋆⋆]

[1] University of Otago, New Zealand
{hans, jruan}@cs.otago.ac.nz
[2] University of Groningen, Netherlands
rineke@ai.rug.nl

Abstract. We model the well-known Sum-and-Product problem in a modal logic, and verify its solution in a model checker. The modal logic is public announcement logic. The riddle is then implemented and its solution verified in the epistemic model checker DEMO.

1 Introduction

The Sum-and-Product problem was first stated—in Dutch—in [1]:

> A says to S and P: I have chosen two integers x, y such that $1 < x < y$ and $x + y \leq 100$. In a moment, I will inform S only of $s = x + y$, and P only of $p = xy$. These announcements remain private. You are required to determine the pair (x, y).
> He acts as said. The following conversation now takes place:
> 1. P says: "I do not know it."
> 2. S says: "I knew you didn't."
> 3. P says: "I now know it."
> 4. S says: "I now also know it."
>
> Determine the pair (x, y).

This problem is, that the agents' announcements *appear* to be uninformative, as they are about ignorance and knowledge and not about (numerical) facts, whereas *actually* they are very informative: the agents learn facts from the other's announcements. For example, the numbers cannot be 14 and 16: if they were, their sum would be 30. This is also the sum of 7 and 23. If those were the numbers their product would have been 161 which, as these are prime numbers, *only* is the product of 7 and 23. So Product (P) would have known the numbers, and therefore Sum (S)—if the sum had been 30—would have considered it possible that Product knew the numbers. But Sum said that he *knew* that Product didn't know the numbers. So the numbers cannot be 14 and 16. Sum and Product learn enough, by eliminations of which we gave an example, to be able to determine the pair of numbers: the unique solution of the problem is the pair $(4, 13)$.

⋆ Hans and Ji appreciate support from AOARD research grant AOARD-05-4017.
⋆⋆ Hans and Rineke appreciate support from the Netherlands Organization for Scientific Research (NWO).

S. Zhang and R. Jarvis (Eds.): AI 2005, LNAI 3809, pp. 790–795, 2005.

Logical approaches to solve the problem are found in [2,3,4,5]. As far as we know, we are the first to use an automated model checker to tackle the Sum-and-Product problem.

In Section 2 we model the Sum-and-Product problem in public announcement logic. In Section 3 we implement the Sum-and-Product specification of Section 2 in DEMO and verify its epistemic features.

2 Public Announcement Logic

Public announcement logic is a dynamic epistemic logic and is an extension of standard multi-agent epistemic logic. Intuitive explanations of the epistemic part of the semantics can be found in [6]. We give a concise overview of the logic.

Language. Given are a set of agents N and a set of atoms Q. The language of public announcement logic is inductively defined as

$$\varphi ::= q \mid \neg\varphi \mid (\varphi \wedge \psi) \mid K_n\varphi \mid C_G\varphi \mid [\varphi]\psi$$

where $q \in Q$, $n \in N$, and $G \subseteq N$ are arbitrary. For $K_n\varphi$, read 'agent n knows formula φ'. For $C_G\varphi$, read 'group of agents G commonly know formula φ'. For $[\varphi]\psi$, read 'after public announcement of φ, formula ψ (is true)'.

Structures. An *epistemic model* $M = \langle W, \sim, V\rangle$ consists of a *domain* W of (factual) *states* (or 'worlds'), *accessibility* $\sim : N \rightarrow \mathcal{P}(W \times W)$, and a *valuation* $V : Q \rightarrow \mathcal{P}(W)$. For $w \in W$, (M, w) is an *epistemic state* (also known as a pointed Kripke model). For $\sim(n)$ we write $\sim_n$, and for $V(q)$ we write V_q.

Semantics. Assume an epistemic model $M = \langle W, \sim, V\rangle$.

$$\begin{array}{ll}
M, w \models q & \text{iff } w \in V_q \\
M, w \models \neg\varphi & \text{iff } M, w \not\models \varphi \\
M, w \models \varphi \wedge \psi & \text{iff } M, w \models \varphi \text{ and } M, w \models \psi \\
M, w \models K_n\varphi & \text{iff for all } \ v \in W : w \sim_n v \text{ implies } M, v \models \varphi \\
M, w \models C_G\varphi & \text{iff for all } \ v \in W : w \sim_G v \text{ implies } M, v \models \varphi \\
M, w \models [\varphi]\psi & \text{iff } M, w \models \varphi \text{ implies } M|\varphi, w \models \psi
\end{array}$$

The group accessibility relation $\sim_G$ is the transitive and reflexive closure of the union of all access for the individuals in G: $\sim_G \equiv (\bigcup_{n \in G} \sim_n)^*$. Epistemic model $M|\varphi = \langle W', \sim', V'\rangle$ is defined as

$$\begin{array}{ll}
W' & = \{w' \in W \mid M, w' \models \varphi\} \\
\sim'_n & = \sim_n \cap (W' \times W') \\
V'_q & = V_q \cap W'
\end{array}$$

The dynamic modal operator $[\varphi]$ is interpreted as an epistemic state transformer. Announcements are assumed to be truthful, and this is commonly known by all agents. Therefore, the model $M|\varphi$ is the model M restricted to all the states

where φ is true, including access between states. The dual of $[\varphi]$ is $\langle\varphi\rangle$: $M, w \models \langle\varphi\rangle\psi$ iff $M, w \models \varphi$ and $M|\varphi, w \models \psi$. Validity and logical consequence are defined in the standard way. For a proof system, see [7].

To give a specification of the Sum-and-Product problem in public announcement logic, first we need to determine the set of atomic propositions and the set of agents. Define $I \equiv \{(x, y) \in \mathbb{N}^2 \mid 1 < x < y \text{ and } x + y \leq 100\}$. Consider the variable x. If its value is 3, we can represent this information as the (truth of) the atomic proposition '$x = 3$'. Slightly more formally we can think of '$x = 3$' as a propositional letter x_3. Thus we create a (finite) set of atoms $\{x_i \mid (i, j) \in I\} \cup \{y_j \mid (i, j) \in I\}$. The set of agents is $\{S, P\}$; S and P will also be referred to as Sum and Product, respectively.

A proposition such as 'Sum knows that the numbers are 4 and 13' is described as $K_S(x_4 \wedge y_{13})$. The proposition 'Sum knows the (pair of) numbers' is described as $K_S(x, y) \equiv \bigvee_{(i,j)\in I} K_S(x_i \wedge y_j)$. Similarly, 'Product knows the numbers' is described as $K_P(x, y) \equiv \bigvee_{(i,j)\in I} K_P(x_i \wedge y_j)$. This is sufficient to formalize the announcements made towards a solution of the problem:

1. P says: "I do not know it": $\neg K_P(x, y)$
2. S says: "I knew you didn't": $K_S\neg K_P(x, y)$
3. P says: "I now know it": $K_P(x, y)$
4. S says: "I now also know it": $K_S(x, y)$

We can interpret these statements on an epistemic model $\mathcal{SP}_{(x,y)} \equiv \langle I, \sim, V\rangle$ consisting of a domain of all pairs $(x, y) \in I$ (as above), with accessibility relations $\sim_S$ and $\sim_P$ such that for Sum: $(x, y) \sim_S (x', y')$ iff $x + y = x' + y'$, and for Product: $(x, y) \sim_P (x', y')$ iff $xy = x'y'$; and with valuation V such that $V_{x_i} = \{(x, y) \in I \mid x = i\}$ and $V_{y_j} = \{(x, y) \in I \mid y = j\}$. The solution of the problem is represented by the truth of the statement

$$\mathcal{SP}_{(x,y)}, (4, 13) \models \langle K_S\neg K_P(x, y)\rangle\langle K_P(x, y)\rangle\langle K_S(x, y)\rangle\top$$

or, properly expressing that $(4, 13)$ is the only solution, by the model validity

$$\mathcal{SP}_{(x,y)} \models [K_S\neg K_P(x, y)][K_P(x, y)][K_S(x, y)](x_4 \wedge y_{13})$$

Note that announcement 1 by Product is superfluous in the analysis. The 'knew' in announcement 2, by Sum, refers to the truth of that announcement in the *initial* epistemic state, not in the epistemic state *resulting* from announcement 1, by Product.

3 The Epistemic Model Checker DEMO

Recently, epistemic model checkers have been developed to verify properties of interpreted systems, knowledge-based protocols, and various other multi-agent systems. The model checkers MCK [8] and MCMAS [9] have a temporal epistemic architecture, and exploration of the search space is based on ordered binary decision diagrams. The epistemic model checker DEMO, developed by Jan van

```
module SNP
where
import DEMO

pairs  = [(x,y)|x<-[2..100], y<-[2..100], x<y, x+y<=100]
numpairs = llength(pairs)
llength [] =0
llength (x:xs) = 1+ llength xs
ipairs = zip [0..numpairs-1] pairs

msnp :: EpistM
msnp = (Pmod [0..numpairs-1] val acc [0..numpairs-1])
  where
  val = [(w,[P x, Q y]) | (w,(x,y))<- ipairs]
  acc = [(a,w,v)| (w,(x1,y1))<-ipairs, (v,(x2,y2))<-ipairs, x1+y1==x2+y2 ]++
        [(b,w,v)| (w,(x1,y1))<-ipairs, (v,(x2,y2))<-ipairs, x1*y1==x2*y2 ]

fmrs1e = K a (Conj [Disj[Neg (Conj [Prop (P x),Prop (Q y)]),
                 Neg (K b (Conj [Prop (P x),Prop (Q y)]))]| (x,y)<-pairs])
amrs1e = public (fmrs1e)
fmrp2e = Conj [(Disj[Neg (Conj [Prop (P x),Prop (Q y)]),
             K b (Conj [Prop (P x),Prop (Q y)]) ] )|(x,y)<-pairs]
amrp2e = public (fmrp2e)
fmrs3e = Conj [(Disj[Neg (Conj [Prop (P x),Prop (Q y)]),
             K a (Conj [Prop (P x),Prop (Q y)]) ] )|(x,y)<-pairs]
amrs3e = public (fmrs3e)

solution = showM (upds msnp [amrs1e, amrp2e, amrs3e])
```

Fig. 1. The DEMO program `SNP.hs`. Comment lines have been removed.

Eijck [10], is not based on temporal epistemics. DEMO is short for Dynamic Epistemic MOdelling. It allows modelling epistemic updates, graphical display of Kripke structures involved, and formula evaluation in epistemic states. DEMO is written in the functional programming language Haskell. The model checker DEMO implements the dynamic epistemic logic of [7]. For a comparative study of these three model checkers, on a different problem, see [11]. We have specified the 'Sum and Product riddle' in DEMO only. The verification of a comparable specification in MCK exceeds its computational power (and this is also to be expected for MCMAS), although clever restriction of variables might well bring such verification with reach.

Figure 1 contains the specification of the Sum and Product riddle in DEMO. The set $I \equiv \{(x, y) \in \mathbb{N}^2 \mid 1 < x < y \text{ and } x+y \leq 100\}$ is realized in DEMO as the list `pairs = [(x,y)| x<-[2..100], y<-[2..100], x<y, x+y<=100]`. A pair such as $(4, 18)$ is not a proper name for a domain element. In DEMO, natural numbers are such proper names. Therefore, we associate each element in `pairs` with a natural number and make a new list `ipairs = zip [0..numpairs-1] pairs`. Here, `numpairs` is the number of elements in `pairs`, and the function

`zip` pairs the i-th element in `[0..numpairs-1]` with the i-th element in `pairs`, and makes that the i-th element of `ipairs`.

The initial model `msnp` of the Sum-and-Product riddle (see Figure 1) is a multi-pointed epistemic model, that consists—this is the line `msnp = (Pmod [0..numpairs-1] val acc [0..numpairs-1])` in the program—of a domain `[0..numpairs-1]`, a valuation function `val`, an accessibility relation function `acc`, and `[0..numpairs-1]` points. As the points of the model are the entire domain, we may think of this initial epistemic state as the (not-pointed) epistemic model underlying it.

The valuation function `val` maps each state in the domain to the subset of atoms that are true in that state. This is different from our previous definition of a valuation V, but the correspondence $q \in \texttt{val}(w)$ iff $w \in V(q)$ is elementary. An element `(w,[P x, Q y])` in `val` means that in state `w`, atoms `P x` and `Q y` are true. For example, given that `(0,(2,3))` is in `ipairs`, `P 2` and `Q 3` are true in state `0`, where `P 2` stands for 'the smaller number is 2' and `Q 3` stands for 'the larger number is 3'. These same facts were described in the previous section by x_2 and y_3, respectively, as that gave the closest match with the original problem formulation. In DEMO, names of atoms *must* start with capital P, Q, R.

The function `acc` specifies the accessibility relations. Agent `a` represents Sum and agent `b` represents Product. For `(w,(x1,y1))` and `(v,(x2,y2))` in `ipairs`, if their sum is the same: `x1+y1==x2+y2`, then they cannot be distinguished by Sum: `(a,w,v)` in `acc`; and if their product is the same: `x1*y1==x2*y2`, then they cannot be distinguished by Product: `(b,w,v)` in `acc`. Function `++` is an operation merging two lists.

Sum and Product's announcements are modelled as structures called 'singleton action models', generated by the announced formula (precondition) φ and an operation `public`. For our purposes it is sufficient to focus on that precondition.

Consider $K_S \neg \bigvee_{(i,j)\in I} K_P(x_i \wedge y_j)$, expressing that Sum says: "I knew you didn't." This is equivalent to $K_S \bigwedge_{(i,j)\in I} \neg K_P(x_i \wedge y_j)$. A conjunct $\neg K_P(x_i \wedge y_j)$ in that expression, for 'Product does not know that the pair is (i,j)', is equivalent to $(x_i \wedge y_j) \rightarrow \neg K_P(x_i \wedge y_j)$. The latter is computationally cheaper to check in the model, than the former: in all states but (i,j) of the model, the latter requires a check on two booleans only, whereas the former requires a check *in each of those states* of Product's ignorance, that relates to his equivalence class for that state, and that typically consists of several states. This explains that the check on $\bigwedge_{(i,j)\in I} \neg K_P(x_i \wedge y_j)$ can be replaced by one on $\bigwedge_{(i,j)\in I}((x_i \wedge y_j) \rightarrow \neg K_P(x_i \wedge y_j))$. Using a model validity, the check on $\bigvee_{(i,j)\in I} K_P(x_i \wedge y_j)$ (Product knows the numbers) can also be replaced, namely by a check $\bigwedge_{(i,j)\in I}((x_i \wedge y_j) \rightarrow K_P(x_i \wedge y_j))$. Using these observations, and writing an implication $\varphi \rightarrow \psi$ as $\neg\varphi \vee \psi$, the three problem announcements 2, 3, and 4 listed on page 792 are checked in DEMO in by the formulas `fmrs1e`, `fmrp2e`, and `fmrs3e`, respectively, as listed in Figure 1. The corresponding singleton action models are obtained by applying the function `public`, for example, `amrs1e = public (fmrs1e)`.

The riddle is solved by updating the initial model `msnp` with the action models corresponding to the three successive announcements:

```
*SNP> showM (upds msnp [amrs1e, amrp2e, amrs3e])
==> [0]
[0]
(0,[p4,q13])
(a,[[0]])
(b,[[0]])
```

This function `showM` displays a pointed epistemic model with, on successive lines, point `[0]`, domain `[0]`—after each update, states are renumbered starting from 0—, valuation `(0,[p4,q13])`—representing the facts `P 4` and `Q 13`, i.e., the solution pair $(4, 13)$—, and accessibility relations `(a,[[0]])` and `(b,[[0]])`—Sum and Product have full knowledge, as their access is the indentity. Intermediate results of the computation can also be given. For the complete output of such interaction, see `www.cs.otago.ac.nz/staffpriv/hans/sumpro/`.

References

1. Freudenthal, H.: (formulation of the sum-and-product problem). Nieuw Archief voor Wiskunde **3(17)** (1969) 152
2. McCarthy, J.: Formalization of two puzzles involving knowledge. In Lifschitz, V., ed.: Formalizing Common Sense : Papers by John McCarthy. Ablex series in artificial intelligence. Ablex Publishing Corporation, Norwood, N.J. (1990) original manuscript dated 1978–1981.
3. Plaza, J.: Logics of public communications. In Emrich, M., Pfeifer, M., Hadzikadic, M., Ras, Z., eds.: Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems. (1989) 201–216
4. Panti, G.: Solution of a number theoretic problem involving knowledge. International Journal of Foundations of Computer Science **2(4)** (1991) 419–424
5. van der Meyden, R.: Mutual belief revision. In Doyle, J., Sandewall, E., Torasso, P., eds.: Proceedings of the 4th international conference on principles of knowledge representation and reasoning (KR), Morgan Kaufmann (1994) 595–606
6. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning about Knowledge. MIT Press, Cambridge MA (1995)
7. Baltag, A., Moss, L., Solecki, S.: The logic of public announcements, common knowledge, and private suspicions. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam (1999) CWI Report SEN-R9922.
8. Gammie, P., van der Meyden, R.: MCK: Model checking the logic of knowledge. In Alur, R., Peled, D., eds.: Proceedings of the 16th International conference on Computer Aided Verification (CAV 2004), Springer (2004) 479–483
9. Raimondi, F., Lomuscio, A.: Verification of multiagent systems via ordered binary decision diagrams: An algorithm and its implementation. In: 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), IEEE Computer Society (2004) 630–637
10. van Eijck, J.: Dynamic epistemic modelling. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam (2004) CWI Report SEN-E0424.
11. van Ditmarsch, H., van der Hoek, W., van der Meyden, R., Ruan, J.: Model checking russian cards. Electronic Notes in Theoretical Computer Science (2005) To appear; presented at MoChArt 05 (Model Checking in Artificial Intelligence).

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
Electronic Notes in Theoretical Computer Science 149 (2006) 105–123
www.elsevier.com/locate/entcs

Model Checking Russian Cards

H.P. van Ditmarsch[1,2]

Computer Science, University of Otago, Dunedin, New Zealand

W. van der Hoek[3]

Computer Science, University of Liverpool, United Kingdom

R. van der Meyden[4]

School of Computer Science and Engineering, University of New South Wales & National ICT Australia, Sydney, Australia

J. Ruan[5]

Computer Science, University of Otago, Dunedin, New Zealand

Abstract

We implement a specific protocol for bit exchange among card-playing agents in three different state-of-the-art epistemic model checkers and compare the results.

Keywords: Cryptography, unconditional security, model checking, information-based protocols, epistemic logic.

doi:10.1016/j.entcs.2005.07.029

1 Introduction

The security of cryptographic protocols generally depends upon the truth of several assumptions: that the agents are computationally limited and that certain computational problems are intractable given these computational limits. In protocols based on public key encryption schemes such as RSA, for example, decryption of messages is tractable for the intended recipient but assumed to be impossible for an eavesdropper, because it requires factoring a large product of primes, a problem assumed to be intractable. There do exist, however, *unconditionally secure* protocols, whose security does not rely upon such assumptions. These protocols can be shown to be secure even against adversaries with unlimited computational powers, because they ensure that the adversary cannot learn secrets for information theoretic rather than computational reasons.

A popular approach to the verification of cryptographic protocols has been to analyse them in terms of information flow as expressed using logics of knowledge and belief [3,12]. In general, the semantics of these logics do not capture the dimension of computational complexity upon which the security of the protocols rest: instead, they treat agents as purely information theoretic reasoners, having computational powers extending even beyond the recursive enumerable. However, this very feature makes these logics a highly appropriate tool for the analysis of unconditionally secure protocols.

In this paper, we consider the application of the logic of knowledge to unconditionally secure protocols based on the exchange of information grounded in correlations arising from a deck of cards having been dealt to the agents. A player can communicate secret bits such as card ownership to another player

[1] This research has been carried out with support from AOARD (Asian Organization for Airforce Research and Development) research grant AOARD-05-4017. National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. Hans van Ditmarsch and Ji Ruan closely collaborated while Ji visited Hans in Otago. Hans thanks Jan van Eijck for the exciting interaction during DEMO's development, and for his suggestions. Wiebe van der Hoek has been involved in 'Russian Cards' matters for a long time, since visiting Otago in 2002 as a William Evans Fellow. Ron van der Meyden wrote the first version of what ultimately became the MCK program, as well as the basis for a first attempt at the MCK program for announcements from arbitrary intiial states, during Hans' visit to Sydney in 2003. Hans and Ji later completed these. Ron later developed much faster versions of both MCK programs. We kindly acknowledge Franco Raimondi's very helpful assistance while completing the MCMAS program. We also thank the anonymous referees supplied by the MoChArt 05 organization for their comments.
[2] Email: hans@cs.otago.ac.nz
[3] Email: wiebe@csc.liv.ac.uk
[4] Email: meyden@cse.unsw.edu.au
[5] Email: jruan@cs.otago.ac.nz

without revealing these secrets to a third player (eavesdropper). This has been investigated in [6,16,10,1,17]. A typical example is the 'Russian Cards Problem': two players each draw three cards from a pack of seven cards, and the remaining player (eavesdropper) gets the last card. The 'problem' is to find protocols that allow the sender and receiver to learn each other's hand of cards, without revealing this information to the eavesdropper. In [16], protocols of length two are presented that solve this problem. Protocols of length greater than two are investigated in [17].

In such card protocols, the required postconditions are not always clear or not easy to verify, publicly known protocol features may involve fairly complex nested dynamic epistemic formulas, and enumeration of all possible protocols is an issue as well. Model checkers are promising tools with which to address these complexities. A model checking analysis has been partially carried out for the Russian Cards problem in [20]: epistemic properties of the scenario are translated into (linear time) LTL, and then verified using the model checker SPIN. A deal of cards together with a number of announcements corresponds to a time line. Uncertainty of the agents is represented by exploiting *local propositions* proposed in [4], see also [13]. This approach to model checking epistemic logic has a number of disadvantages: the need for translation means that the epistemic aspects are only implicit in the analysis, it requires that the appropriate local propositions – which may be difficult to identify – be explicitly provided by the user, and in the case of negative occurrences of the knowledge operator, multiple runs of the model checker are necessary to conduct the verification.

In this contribution, we take a more direct approach, verifying protocol properties in model checkers which work with epistemic logic explicitly. We conduct a comparative study of a number of systems, based on a variety of approaches to representing the evolution of knowledge: combinations of the logic of knowledge with linear and/or branching time [5,9,14], and dynamic epistemic logics [8,2,15]. Specifically, we consider the model checkers MCK [7] which deals with the logic of knowledge and both linear and branching time using BDD based algorithms, MCMAS [11] which handles knowledge and branching time using BDD based algorithms, and DEMO [18], which is an explicit state model checker based on a dynamic epistemic logic.

We have selected one specific Russian Cards protocol, the 'five hands protocol', implemented it in these quite different dedicated 'epistemic' model checkers, and verified its relevant properties. This involved reinterpreting dynamic epistemic concepts in temporal epistemic terms; this theoretical exercise was carried out successfully and increased our understanding of dynamic epistemic features. All three implementations were carried out within a rea-

sonable development time and all were successful. Some additional Russian Cards protocol features, in particular for protocols of length greater than two, have been kept outside this comparison. Also, incorrect protocols (such as for non-solutions of the Russian Cards problem) can be easily shown to be so by establishing failure of (commonly known or other) epistemic conditions. This only requires (almost) trivial changes in the scripts presented below for a correct protocol.

In Section 2 we present the Russian Cards problem. Sections 3 to 5 are dedicated to the implementation of the 'five hands' protocol for the Russian Cards problem in the model checkers, respectively, MCK, DEMO, and MCMAS. Section 6 compares the results. The MCK, DEMO, and MCMAS input scripts can be found on `http://www.cs.otago.ac.nz/staffpriv/hans/aoard/`.

2 Russian Cards

From a pack of seven known cards two players each draw three cards and a third player gets the remaining card. How can the players with three cards openly inform each other about their cards, without the third player learning from any of their cards who holds it?

This 'Russian Cards' problem originated at the Moscow Math Olympiad 2000. Call the players Anne, Bill and Cath, and the cards $0, ..., 6$, and suppose Anne holds $\{0, 1, 2\}$, Bill $\{3, 4, 5\}$, and Cath card 6. For the hand of cards $\{0, 1, 2\}$, write 012 instead, for the card deal, write 012.345.6, etc. Assume from now on that 012.345.6 is the actual card deal. All announcements must be public and truthful. There are not many things Anne can safely say. Obviously, she cannot say "I have 0 or 6," because then Cath learns that Anne has 0. But Anne can also not say "I have 0 or 3," because Anne does not know if Cath has 3 or another card, and *if* Cath had card 3, she would have learnt that Anne has card 0. But Anne can also not say "I have 0 or 1." Even though Anne holds both 0 and 1, so that she does not appear to risk that Cath eliminates either card and thus gains knowledge about single card ownership (weaker knowledge, about alternatives, is allowed), Cath knows that Anne will not say anything from which Cath may learn her cards. And thus Cath can conclude that Anne will only say "I have 0 or 1" if she actually holds both 0 and 1. And in that way Cath learns two cards at once! The apparent contradiction between Cath not knowing and Cath knowing is not really there, because these observations are about different information states: it is merely the case that announcements may induce further updates that contain yet other information.

Whenever after Anne's announcement it is (at least) not common knowl-

edge to Anne, Bill, and Cath, that Cath remains ignorant of any of Anne's or Bill's cards, this may be informative to Cath after all. A typical example is when *Anne says that she either holds 012 or not any of those cards, after which Bill says that Cath holds card 6.* For details, see [16]. Indeed, a solution requirement is that Cath's ignorance remains public knowledge after any announcement. Such announcements are called *safe*.

A solution to the Russian Cards problem is a sequence of safe announcements after which it is commonly known to Anne and Bill (not necessarily including Cath) that Anne knows Bill's hand and Bill knows Anne's hand. This (instance of a) five hands protocol is a solution:

> *Anne says "My hand of cards is one of 012, 034, 056, 135, 246," after which Bill says "Cath has card 6."*

Note that Bill's announcement is equivalent to "My hand of cards is one of 345, 125, 024." After this sequence, it is even publicly known that Anne knows Bill's hand and Bill knows Anne's hand. If we extend Anne's announcement with one more hand, namely 245, and if it is public knowledge that the protocols used by Anne and Bill are of finite length (so may consist of more than two announcements), then it is 'merely' common knowledge to Anne and Bill that they know each other's hand, but (disregarding further analysis) Cath considers it possible that they do not know each other's hand of cards. This is a useful security feature for Anne and Bill, as Cath plays the role of the eavesdropper. A further postcondition is that all safe announcements by Anne ensure at least one safe response from Bill, and vice versa. This recursive requirement results in a more complex condition. See [17].

Public announcement logic The Russian Cards problem can be modelled in public announcement logic with common knowledge. We give a concise overview of the language and its semantics.

Given are a set of agents N and a set of atoms P. An *epistemic model* $M = \langle S, \sim, V \rangle$ consists of a *domain* S of (factual) *states* (or 'worlds'), *accessibility* $\sim : N \to \mathcal{P}(S \times S)$, and a *valuation* $V : P \to \mathcal{P}(S)$. For $s \in S$, (M, s) is an *epistemic state*. For $\sim(n)$ we write $\sim_n$, and for $V(p)$ we write V_p. So, access $\sim$ can be seen as a set of equivalence relations $\sim_n$, and V as a set of valuations V_p. For $(\bigcup_{n \in G} \sim_n)^*$, write $\sim_G$: this is access to interpret common knowledge for group G.

The language of public announcements is inductively defined as

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid K_n\varphi \mid C_G\varphi \mid [\varphi]\psi$$

where $p \in P$, $n \in N$, and $G \subseteq N$ are arbitrary. For $K_n\varphi$, read 'agent n

knows formula φ'. For $C_G\varphi$, read 'group of agents G commonly know formula φ'. For $[\varphi]\psi$, read 'after public announcement of φ, formula ψ (is true)'. The effect of the public announcement of φ is the restriction of the epistemic state to all worlds where φ holds. So, 'announce φ' can be seen as an information state transformer, with a corresponding dynamic modal operator $[\varphi]$.

The semantics is as follows. Given is an epistemic model $M = \langle S, \sim, V\rangle$.

$$\begin{array}{ll} M, s \models p & \text{iff } s \in V_p \\ M, s \models \neg\varphi & \text{iff } M, s \not\models \varphi \\ M, s \models \varphi \wedge \psi & \text{iff } M, s \models \varphi \text{ and } M, s \models \psi \end{array}$$

$$\begin{array}{l} M, s \models K_n\varphi \text{ iff for all } \ t \in S : s \sim_n t \text{ implies } M, t \models \varphi \\ M, s \models C_G\varphi \text{ iff for all } \ t \in S : s \sim_G t \text{ implies } M, t \models \varphi \\ M, s \models [\varphi]\psi \text{ iff } M, s \models \varphi \text{ implies } M|\varphi, s \models \psi \end{array}$$

Model $M|\varphi = \langle S', \sim', V'\rangle$ is defined as

$$\begin{array}{ll} S' & \equiv \{s' \in S \mid M, s' \models \varphi\} \\ \sim'_n & \equiv \sim_n \cap \, (S' \times S') \\ V'_p & \equiv V_p \cap S' \end{array}$$

In other words: the model $M|\varphi$ is the model M restricted to all the states where φ holds, including access between states. Formula φ is valid on model M, notation $M \models \varphi$, if and only if for all states s in the domain of M: $M, s \models \varphi$. Formula φ is valid, notation $\models \varphi$, if and only if for all models M: $M \models \varphi$.

We now model the Russian Cards problem in this logic. Given a stack of known cards and some players, the players blindly draw some cards from the stack. In a state where cards are dealt in that way, but where no game actions of whatever kind have been done, it is commonly known what the cards are, that they are all different, how many cards each player holds, and that players only know their own cards. From the last it follows that two deals are *the same for an agent*, if she holds the same cards in both, and if all players hold the same number of cards in both. This induces an equivalence relation on deals.

An epistemic model $(Rus, 012.345.6)$ for the deal 012.345.6 that we investigate, encodes the knowledge of the players Anne, Bill and Cath (a, b, c) in

this card deal. It consists of $\binom{7}{3}\binom{4}{3}\binom{1}{1} = 140$ deals. For each player, access between states is induced by the equivalence above, for example, $012.345.6 \sim_a 012.346.5$ says that Anne cannot tell these two card deals apart (as her hand is 012 in both). Facts about card ownership written as q_n, for 'card q is held by player n'. The valuation V_{0_a} of fact 0_a (Anne holds card 0) consists of all 60 deals where 0 occurs in Anne's hand, etc.

After a sequence of announcements that is a solution of the Russian Cards problem, it should hold that Anne knows Bill's cards, that Bill knows Anne's cards, and that Cath doesn't know any of Anne's or Bill's cards:

$$\begin{aligned} \mathsf{a\_knows\_bs} &\equiv \bigwedge_{q=0..6}(K_a q_b \vee K_a \neg q_b) \\ \mathsf{b\_knows\_as} &\equiv \bigwedge_{q=0..6}(K_b q_a \vee K_b \neg q_a) \\ \mathsf{c\_ignorant} &\equiv \bigwedge_{q=0..6}(\neg K_c q_a \wedge \neg K_c q_b) \end{aligned}$$

We suggested in the previous section that these conditions are too weak. This can be exemplified by the observation that, e.g.,

$$Rus, 012.345.6 \models [K_a(012_a \vee (\neg 0_a \wedge \neg 1_a \wedge \neg 2_a))][\mathsf{c\_ignorant}]\neg\mathsf{c\_ignorant}$$

After Anne says that her hand is 012 or that she does not hold any of those cards, $\mathsf{c\_ignorant}$ is true, but a further update with that (in other words: when Cath can assume that this is true) makes Cath learn some of Anne's cards, so that $\mathsf{c\_ignorant}$ is false. The actually required postconditions avoiding such complications are: after every announcement of an executed protocol, it is publicly known that Cath is ignorant, and after the execution of the entire protocol it is commonly known to Anne and Bill that: Anne knows that Bill knows her hand of cards, and Bill knows that Anne knows his hand of cards. Also using that $C_{ab}(K_b\mathsf{a\_knows\_bs} \wedge K_a\mathsf{b\_knows\_as})$ is equivalent to $C_{ab}(\mathsf{a\_knows\_bs} \wedge \mathsf{b\_knows\_as})$ this is formalized as

$$\begin{aligned} &C_{ab}(\mathsf{a\_knows\_bs} \wedge \mathsf{b\_knows\_as}) \\ &C_{abc}\mathsf{c\_ignorant} \end{aligned}$$

Concerning protocols: when Anne announces 'φ', this should be interpreted as '$K_a\varphi$' given that she knows what she says, and even as '$K_a\varphi \wedge [K_a\varphi]K_a\mathsf{c\_ignorant}$' given her intention, and beyond that even as '$K_a\varphi \wedge [K_a\varphi]C_{abc}K_a\mathsf{c\_ignorant}$' given that her intention is public. One can then show that

$$Rus, 012.345.6 \models [K_a\varphi \wedge [K_a\varphi]C_{abc}\mathsf{c\_ignorant}]C_{abc}\mathsf{c\_ignorant}$$

So in this case the intention is indeed realized, unlike above: the announcement is safe. We ignore the further complication that safe announcements require safe responses in this submission. The solution given in Section 2 consists of the successive announcements

$$\mathsf{a\_announce} \equiv 012_a \vee 034_a \vee 056_a \vee 135_a \vee 246_a$$

$$\mathsf{b\_announce} \equiv 6_c$$

Temporal epistemic logics Two of the model checkers that we present require interpreted system representations and / or verification of temporal epistemic logical formulas. Therefore, some words are in order on how these compare to a dynamic epistemic setting.

An *interpreted system* $\mathcal{I}$ is a pair $(\mathcal{G}, \mathcal{R})$ consisting of a set of global states $\mathcal{G}$ and a set of runs $\mathcal{R}$ relating those states. A *global state* $g \in \mathcal{G}$ is a tuple consisting of local states g_n for each agent and a state g_ϵ of the environment. A *run* $r \in \mathcal{R}$ is a sequence of global states. The m-th global state occurring in a run r is referred to as $r(m)$, and the local state for agent n in a global state $r(m)$ is written as $r_n(m)$.

A *point* (r, m) is a pair consisting of a run and a point in time m – this is the proper abstract domain object when defining epistemic models for interpreted systems. In an interpreted system, agents can distinguish global states from one another iff they have the same local state in both, which induces

$$(r, m) \sim_n (r', m') \text{ iff } r(m) \sim_n r'(m') \text{ iff } r_n(m) = r'_n(m')$$

With the obvious valuation for local and environmental state values, that defines an epistemic model. For convenience we keep writing $\mathcal{I}$ for that. Given an actual point (r', m'), we thus get an epistemic state $(\mathcal{I}, (r', m'))$. Epistemic and (LTL) temporal (next) operators have the interpretation

$$\mathcal{I}, (r, m) \models X\varphi \quad \text{iff} \quad \mathcal{I}, (r, m+1) \models \varphi$$

$$\mathcal{I}, (r, m) \models K_n\varphi \quad \text{iff} \quad \text{for all } (r', m') : (r, m) \sim_n (r', m') \text{ implies } \mathcal{I}, (r', m') \models \varphi$$

We now *outline* the relation between 'next' and announcement operators. An announcement is seen as a completely observable clock tick, synchronizing the system. Announcing φ at time m is simulated in $\mathcal{I}$ by changing the value of some environmental variable p for exactly those points where φ is true, when transiting from point (r, m) to point $(r, m+1)$, and passing on that information to the local states of the agents. The static information available at time m is contained in the restriction $\mathcal{I}|m$ of the interpreted system $\mathcal{I}$ to all points for

time m. This determines the meaning of purely epistemic formulas. But for formulas containing epistemic and 'next'-temporal operators the situation is more complex. Assume that for each time m there is a formula φ such that the *only* transitions allowed at m are those induced by announcement of φ. We can define a translation $*$ where, given an epistemic state and a formula, each X-operator in that formula is replaced by a corresponding dynamic operator $[\varphi]$. The following now are all equivalent

$$\begin{aligned}
&\text{if } \mathcal{I}, (r,m) \models \varphi, \text{ then } \mathcal{I}, (r,m) \models X\psi \\
&\text{if } \mathcal{I}, (r,m) \models \varphi, \text{ then } \mathcal{I}, (r,m+1) \models \psi \\
&\text{if } \mathcal{I}|m, (r,m) \models \varphi^*, \text{ then } \mathcal{I}|m|\varphi^*, (r,m) \models \psi^* \\
&\mathcal{I}|m, (r,m) \models [\varphi^*]\psi^*
\end{aligned}$$

In case φ and ψ are both purely epistemic, so that $\varphi^* = \varphi$, and $\psi^* = \psi$, we have that

$$\mathcal{I}, (r,m+1) \models \psi \text{ corresponds to } \mathcal{I}|m|\varphi, (r,m) \models \psi$$

It is interesting to observe, that checking ψ in the former involves (given synchronous perfect recall) the *entire* domain of $\mathcal{I}|(m+1)$, whereas checking ψ in the latter *only* involves the φ-states of its predecessor $\mathcal{I}|m$, corresponding to *only* one value of the environmental variable that is reset in the transition from m to $m+1$. For 'Russian Cards', the first announcement reduces the domain from 140 to 20 points, and the second from 20 to 3 points.

3 Model Checker MCK

MCK, for 'Model Checking Knowledge', is a prototype model checker for temporal and knowledge specifications, developed by Peter Gammie and Ron van der Meyden [7]. The overall setup supposes a number of agents acting in an environment, by temporal development. This is modelled by an interpreted system where agents perform actions according to a protocol. Actions and the environment may be only partially observable at each instant in time.

Different approaches to the temporal and epistemic interaction and development are implemented. Knowledge may be based on current observations only, on current observations and clock value, and on the history of all observations and clock value. The last corresponds to synchronous perfect recall. We have used that approach. In the temporal dimension, the specification formulas may describe the evolution of the system along

a single computation, i.e., using linear time temporal logic (LTL), or they may describe the branching structure of all possible computations, i.e, using branching time or computation tree logic (CTL). We have used LTL. See http://www.cse.unsw.edu.au/~mck/ for more information.

Russian Cards in MCK In MCK, we have to reinterpret the dynamic epistemics of Section 2 in temporal epistemic terms. In a program `rus.mck` we successively introduce environmental variables and initialize those; we create three agents `A`, `B`, and `C` with corresponding protocols `"anne"`, `"bill"` and `"cath"`; a main part of the program specifies the (temporal) transitions, induced by card dealing and the announcements, that relate different information states for these players; finally `rus.mck` contains a part with various to be verified properties of the timelines created.

A hand of cards of an agent is encoded by a list of seven booleans, for example `a_hand : Bool[7]` specifies for all of the cards 0, ..., 6 whether they are held by Anne or not, such that `anne_cards[0]` is true when Anne holds card 0, etc. Initially, such variables are set to false.

Agent A, for Anne, is created by

```
agent A "anne" (a_hand, a_announce, b_announce, stage)
```

The name of the agent is `A`. It uses protocol `"anne"`. It can interact with, and potentially observe the variables between parentheses. The first of those is, obviously, only observable by Anne, the others will reappear in the other agent definitions, as they are publicly observable. The variable `stage` is the 'clock tick'.

The `transitions` part of `rus.mck` specify what happens in different stages of the execution of the protocol. We distinguish stages (clock ticks) 0, 1, 2, and 3. In stage 0 the cards are dealt to the players, in the order 0, ..., 6. We show it up to the dealing of card 0.

```
 stage == 0  ->
   begin  if
             na < 3  -> begin a_hand[0]:=True; na:= na+1 end []
             nb < 3  -> begin b_hand[0]:=True; nb:= nb+1 end []
             nc == 0 -> begin c_hand[0]:=True; nc:= 1 end
          fi;
```

Variables `na`, `nb`, and `nc` are counters to record how many cards agents have, and `[]` means nondeterministic choice. In this part of the transitions, 140 different deals are created, represented as 140 different timelines.

In stage 1, Anne announces that her hands is one of 012, 034, 056, 135, and 246. This is done indirectly by executing the protocol `"anne"`, that contains a

condition corresponding to these five deals, which causes the action `Announce` to be executed. This then results in the atom `a_announce` becoming true.

```
stage == 1 /\ A.Announce -> a_announce := True
```

In stage 2, Bill announces that Cath holds card 6. Alternatively, one can model that Bill announces Cath's card – whatever it is. Bill's announcement is by way of an action `B.Announce`, and results in the variable `b_announce` to become true. This is the transition to stage 3, the final stage. We can imagine the whole system to consist of 140 different runs. Whether variables `a_announce` and `b_announce` are true in stage 2 and stage 3, respectively, depends on the deal in that run.

The protocol for Anne is

```
protocol "anne" (cards: observable Bool[7],
      a_announce: observable Bool, b_announce: observable Bool,
      stage: observable Counter)
begin
  skip; if
   ( (cards[0] /\ cards[1] /\ cards[2]) \/
     (cards[0] /\ cards[3] /\ cards[4]) \/
     (cards[0] /\ cards[5] /\ cards[6]) \/
     (cards[1] /\ cards[3] /\ cards[5]) \/
     (cards[2] /\ cards[4] /\ cards[6]) )
    -> <<Announce>>
  fi
end
```

The 'begin-end' part of this protocol specifies for each of the stages 0, 1, and 2 what happens in that stage. In stage 0 nothing happens: `skip`. In stage 1, the action `Announce` – that is, whatever is found between << and >> – is executed. Actually, the value or instance of `cards` for Anne is `a_cards`; see above, where Anne is created. Alternatively to five actual hands, a much longer protocol creates five arbitrary hands of cards based on Anne's actual hand. Nothing is specified for stage 2: this is therefore `skip` again by default. Bill has a similar protocol but his protocol starts with `skip ; skip`, as his announcement is in stage 2. And Cath does not act at all, which carries the protocol `skip ; skip ; skip`.

The knowledge of the agents evolves with every stage, via the agents' limited access to the environment. Initially, they only observe their own hand of cards, and Anne's and Bill's public announcement is accessed by all agents. Anne cannot distinguish two states iff her observations are the same in those states. For example, in stage 1 Anne cannot distinguish the timelines for deals

012.345.6 and 012.346.5, because: both have the same `a_hand` values (for all seven variables), `a_announce` is true in both cases, and `b_announce` is false is both cases. But in stage 3, Anne can distinguish these timelines, since `b_announce` is true for the former and false for the latter.

A final part of `rus.mck` lists various temporal epistemic properties to be checked. For example, we want to verify that $Rus, 012.345.6 \models [\mathsf{a\_announce}][\mathsf{b\_announce}]C_{ab}\mathsf{a\_knows\_bs}$. The current version (0.2.0) of MCK does not support common knowledge operators for specification in the perfect recall module. Therefore we verify instead that in stage 3, $\mathsf{a\_knows\_bs}$ is valid in the model. This corresponds to $Rus|\mathsf{a\_announce}|\mathsf{b\_announce} \models \mathsf{a\_knows\_bs}$ which ensures that $Rus|\mathsf{a\_announce}|\mathsf{b\_announce}, 012.345.6 \models C_{abc}\mathsf{a\_knows\_bs}$. And in this specific model $C_{ab}\mathsf{a\_knows\_bs} \leftrightarrow C_{abc}\mathsf{a\_knows\_bs}$ is also valid.

```
spec_spr_xn = X 3 ( (a_announce /\ b_announce) =>
  (  (((Knows A b_hand[0]) \/ (Knows A neg b_hand[0]))) /\
              (...)
     (((Knows A b_hand[6]) \/ (Knows A neg b_hand[6]))) ))
```

The part `spec_spr_xn` means that we are using the perfect recall module of MCK, and `X 3` is the triple 'next state' temporal operator, counting from stage 0. Therefore, the formula bound by the operator is checked in stage 3. Similarly, other properties of the five hands protocol are verified.

4 Model Checker DEMO

The tool DEMO is developed by Jan van Eijck [18]. DEMO is short for Dynamic Epistemic MOdelling. It allows modelling epistemic updates, graphical display of Kripke structures involved (i.e., epistemic or state models, and action models that represent epistemic actions), formula evaluation in epistemic states, etc. Epistemic models are minimized under bisimulation, and action models are minimized under the (more appropriate, weaker) notion of action emulation [19]. DEMO is written in the functional programming language Haskell. See also http://www.cwi.nl/~jve/papers/04/demo/.

The model checker DEMO implements the dynamic epistemic logic of [2]. In this 'action model logic' the global state of the multi-agent system is represented by an epistemic model (multi-agent Kripke model), and the agents' action is represented by an action model. An action model is also based on a multi-agent Kripke frame, but instead of carrying a valuation it has a precondition function which assigns a precondition to each point in the action model, which stands for an atomic action. The state change in the system is via an operation called update product. This is a restricted modal product. In this submission we restrict our attention to action models for public an-

nouncements. Such action models have a singleton domain. We refrain from details and proceed with the recursive definition of formulas in DEMO.

```
Form = Top | Prop Prop | Neg  Form | Conj [Form] | Disj [Form]
         | K Agent Form | CK [Agent] Form
```

Formula `Top` stands for $\top$, `Prop Prop` for atomic propositional letters (the first occurrence of `Prop` means that the datatype is 'propositional atom', whereas the second occurrence of `Prop` is the placeholder for an actual proposition letter, such as `P0`), `Neg` for negation, `Conj [Form]` stands for the conjunction of a list of formulas of type `Form`, similarly for `Disj`, `K Agent` stands for the individual knowledge operator for agent `Agent`, and `CK [Agent]` for common knowledge operator for the group of agents listed in `[Agent]`.

A pointed (and singleton) action model for a public announcement is created by a function `public` with a precondition (formula) as argument. The update operation is specified as

```
upd ::  EpistM -> PoAM -> EpistM
```

Here, `EpistM` is an epistemic state and `PoAM` is a pointed action model. The update generates a new epistemic state as specified above. Formula checking is defined as

```
isTrue :: EpistM -> Form -> Bool
```

Its arguments are an epistemic state and a formula, and it returns a boolean value.

Russian Cards in DEMO In DEMO, one is restricted to propositional letters starting with lower case p, q and r, so we cannot write, for example, 0_a for the atomic proposition that Anne holds card 0, as in Section 2. Instead, atoms {`p`, ..., `p6`, `q`, ..., `q6`, `r`, ..., `r6`} represent such atomic propositions. The name `p4` – Anne holds card 4 – actually stands for `Prop (P 4)`, etc. Instead of `p0` we write, somewhat arbritrarily, `p`, and similarly for `q` and `r`.

The initial epistemic state `rus` representing the knowledge in card deal 012.345.6 is constructed as follows. A set of integers `[0..139]` represents the 140 different deals. Each integer is associated with seven propositional letters – the valuation of facts in that state. The first two deals correspond to the valuations

```
(0,[P 0,P 1,P 2,Q 3,Q 4,Q 5,R 6]),
(1,[P 0,P 1,P 2,Q 3,Q 4,Q 6,R 5])
```

The deal numbered 0 stands for actual deal 012.345.6. A pair of two integers is in the accessibility relation for an agent n, if that agent holds the same cards in

both deals. Two such pairs for Anne are (a,0,0),(a,0,1). DEMO assumes arbitrary accessibility relations. So, unfortunately, we have to explicitly list all pairs in the equivalence relation for each agent, as above.

Anne's public announcement a_announce corresponds to the following singleton action model named a_announce, which is produced by the function public.

```
public( K a (Disj[Conj[p,p1,p2],Conj[p,p3,p4],Conj[p,p5,p6],
             Conj[p1,p3,p5],Conj[p2,p4,p6]]))
```

Similarly, we have an action model b_announce for Bill's announcement b_announce. The postcondition that Anne knows Bill's hand of cards, a_knows_bs, is represented as

```
aknowsbs = Conj[ Disj[K a q,  K a (Neg q)  ],
                 Disj[K a q1, K a (Neg q1) ],
                 Disj[K a q2, K a (Neg q2) ],
                 Disj[K a q3, K a (Neg q3) ],
                 Disj[K a q4, K a (Neg q4) ],
                 Disj[K a q5, K a (Neg q5) ],
                 Disj[K a q6, K a (Neg q6) ]  ]
```

Similarly for b_knows_as and c_ignorant. The model checker now verifies the postconditions of the constructed models. After Bill's announcement it is common knowledge to Anne and Bill that Anne knows Bill's hand of cards, and it is also common knowledge to Anne and Bill that Bill knows Anne's hand of cards. It is publicly known that Cath remains ignorant:

```
*RUS>isTrue (upd (upd rus a_announce) b_announce) (CK [a,b] a_knows_bs)
True
*RUS>isTrue (upd (upd rus a_announce) b_announce) (CK [a,b] b_knows_as)
True
*RUS>isTrue (upd (upd rus a_announce) b_announce) (CK [a,b,c] c_ignorant)
True
```

Epistemic state (upd rus a_announce) is the result of updating the initial epistemic state rus with singleton pointed action model with precondition a_announce – to improve readability we have chosen to name the action model a_announce and not the precondition. The epistemic state (upd (upd rus a_announce) b_announce) is the result of updating epistemic state (upd rus a_announce) with the singleton pointed action model named b_announce (with that precondition).

5 Model Checker MCMAS

MCMAS presumably stands for Model Checking Multi-Agent Systems. This model checker has been developed by Franco Raimondi and Alessio Lomuscio [11]. The current version is `mcmas 0.6`. System descriptions and protocol properties are verified using ordered binary decision diagrams, comparable to the approach used in MCK. It extends existing obdd-based techniques for reactive systems by adding both an epistemic (ATL) and a deontic dimension to the logical language, and allowing input in terms of interpreted systems. MCMAS is implemented in C^{++}. For more information, see http://www.cs.ucl.ac.uk/staff/F.Raimondi/MCMAS/.

In MCMAS, the global state is represented as a tuple of the local states of the agents. For Russian Cards, agents `Anne`, `Bill`, and `Cath` represent players, and an agent `Env` (the environment) represents the card deal. The local state of agent Anne requires five components, that can be seen as variables; three represent her hand of cards, and two the status quo and outcome of the two announcements. Version 0.6 of MCMAS does not support variables in the description of agents' local states. Therefore we encode the variable parts in a single string. For example, one local state for Anne is `a012tf`. This means that Anne holds cards 0,1, and 2, that Anne's announcement a_announce has been (truthfully) made in the global state of which this local state is a component, and that Bill's announcement b_announce could not be made (was false) in that global state. Similarly, we have five variables for Bill, and three variables for Cath. The local state of the agent `Env` has seven variables, because it represent a card deal. An example is `e0123456`. This stands for the actual deal 012.345.6.

The information changes take the usual steps: (1) the cards are revealed to the agents, (2) Anne announces a_announce, and (3) Bill announces b_announce. All reachable global states will be included in the next stage. An example initial global state is `(annnnn, bnnnnn, cnnn, e0123456)`; an '`n`' essentially means that the agent has no information on the value of corresponding variable, modelled by giving the variable that value `n`. So, `bnnnnn` means that Bill's local state is that he does not know his cards yet (the first three n's), that Anne has not made her announcement yet (the fourth n) and that Bill has not made his announcement yet. The above global state `(annnnn, bnnnnn, cnnn, e0123456)` then transits to `(a012nn, b345nn, c6nn, e0123456)`, where each agent knows what cards it holds. Anne's a_announce is then made, causing the transition to `(a012tn, b345tn,c6tn, e0123456)` and b_announce finally results in `(a012tt, b345tt,c6tt, e0123456)` – this time, Bill's announcement is successful. These state transitions are specified in the program. For example, for agent Anne, the transition for step one is as follows; `Lstate` is

the local state of (current) agent `Anne`, and `Env.Lstate` is the local state of `Env`.

```
a012nn if (Lstate=annnnn and
    ( Env.Lstate=e0123456 or Env.Lstate=e0123465  or
      Env.Lstate=e0123564 or Env.Lstate=e0124563 ));
```

The environment `Env` does not change during transitions, but this has to be made explicit as

```
e0123456 if Lstate=e0123456;
```

In the 'valuation' part of an MCMAS program we define what can be seen as (the denotation of) atomic propositions. For example

```
ab_d0123456 if (Anne.Lstate=a012tt and Bill.Lstate=b345tt and
               Cath.Lstate=c6tt and Env.Lstate=e0123456);
```

is the atom that is (uniquely) true in the global state `(a012tt, b345tt,c6tt, e0123456)`. Similarly, atoms expressing card ownership such as 0_a for 'Anne holds card 0' are defined by enormous expressions starting as (and consisting of 60 alternative card deals)

```
a0 if (Env.Lstate=e0123456 or Env.Lstate=e0123465 or ...
```

Groups of agents can be named too. This is useful when checking common knowledge. For example, expression `ABC={Anne, Bill, Cath};` gives the group consisting of Anne, Bill, and Cath the label `ABC`. The common knowledge formula $C_{abc}(0_a \rightarrow K_a 0_a)$ is then represented as `CK(ABC,a0->K(Anne,a0))`. We conclude this short exposition with the postcondition C_{abc}c_ignorant that verifies that Cath remains ignorant after both announcements have been made – '!' stands for negation.

```
ab_d0123456 -> GCK(ABC,(
  ( !K(Cath,a0) and !K(Cath,b0) ) and
  ( !K(Cath,a1) and !K(Cath,b1) ) and
  ( !K(Cath,a2) and !K(Cath,b2) ) and
  ( !K(Cath,a3) and !K(Cath,b3) ) and
  ( !K(Cath,a4) and !K(Cath,b4) ) and
  ( !K(Cath,a5) and !K(Cath,b5) ) and
  ( !K(Cath,a6) and !K(Cath,b6) ) ));
```

6 Comparison

Rough performance results for the input scripts described above are based on a PC configuration Linux 2.4.30 i686 Pentium 4, 800Mhz and 2018M RAM.

The times required, respectively, for the Russian Cards five hands protocol, as an average over five runs, are:

- MCK – 160 seconds (Long BDD package) or 109 seconds (CUDD BDD package)
- MCMAS – 117 seconds (CUDD BDD package)
- DEMO – 9 seconds

The time measure for MCK and MCMAS is for the whole model checking process, i.e., both model construction and formula checking. For MCMAS it includes the time to autogenerate the MCMAS input script from a C program. DEMO operates on slightly different principles: First, the Haskell interpreter compiles RUS.hs and related modules DPLL and DEMO. Only then, we check individual formulas. We measured the combined autogeneration, compilation and checking steps.

These results cannot be straightforwardly interpreted as indicative of the relative performance of the model checkers, however, as they are based on rather different modellings and model checking questions. One difference is that the MCK input script explicitly represents the dealing of cards using a transition program, whereas the input to MCMAS and DEMO already have the results of a deal explicitly represented in the initial states. Another is that MCK and MCMAS check a temporal property for *all* initial states, whereas DEMO checks a dynamic property at a *single* initial state. The runtimes can also be quite sensitive to specific choices made in the modelling. Apart from the scripts discussed in this contribution, we later developed a *much* more concise DEMO program, as well as an alternate MCK modelling in which the dealing of cards is represented by a constraint on initial states rather than by a program. We refrain from details and refer instead to the companion website. The complexity results for these versions are

- DEMO-new – 4 seconds
- MCK-new – 1.1 seconds (Long BDD), 0.27 seconds (CUDD)

The modellings discussed above focus on announcements for the specific situation of the deal 012.345.6. We have also developed an MCK script modelling a protocol that provides an five hands announcement for Anne for an *arbitrary* initial state. This script currently requires about 3 hours to run, and is still a subject of our experiments.

Mostly, however, we were interested in how versatile the tools appeared to be, to implement a problem that was originally formulated in local, and dynamic epistemic, terms, into temporal epistemic terms and/or as an inter-

preted system. In other words, we were more than anything else interested in development time and supported functionality. Conclusions based on our experiences are extremely tentative. Implementing the Russian Cards problem in DEMO took about half a day, for Ji Ruan, who is an expert in DEMO. MCK scripts developed by Ron van der Meyden, expert in MCK, also took about half a day. Currently, MCK does not support common knowledge (in the used module), nor epistemic preconditions, nor preconditions to temporal formulas. The last makes it impossible to have knowledge preconditions to players' announcements. Such preconditions are *always* epistemic, as agents only announce what they know to be true. Also, unsuccessful updates – formulas that become false because they are announced – cannot be made visible in the way they have to be checked in MCK: the analogue is a conditional formula where the antecedent is also a subformula of the temporal consequent. On the other hand, MCK allows a very natural formalization of protocols – this is not, or less, possible in DEMO or MCMAS. The 'fully interpreted system' approach of MCMAS is very transparent, but the models that need to be built are 'very' large: (automated input of) thousands of lines of code, as opposed to (manual input of) about a hundred lines of code in MCK. More than anything, this case-study increased our insight into the state of the art in epistemic model checking, and our understanding of the theoretical issues involved in card cryptography, emerging from the need to reformulate these issues in different logics.

7 Conclusions

We have implemented the five hands protocol to solve the Russian Cards problem in the model checkers MCK, DEMO, and MCMAS. Dynamic epistemic requirements can be easily reformalized in temporal epistemic terms, a necessary requirement for formalization in MCK and MCMAS. The model checkers vary in how easy, or difficult, it is to build the initial epistemic state, in how difficult it is to formalize announcements and execute them in that initial state, and in how to verify protocol properties. We intend to pursue this investigation by implementing more complex protocols and verifying more complex properties for such 'card cryptography', and generalize it to the level of interpreted systems with agent dependencies, where groups of agents aim to share their local state value while keeping it a secret from the remaining agents.

References

[1] M.H. Albert, R.E.L. Aldred, M.D. Atkinson, H.P. van Ditmarsch, and C.C. Handley. Safe communication for card players by combinatorial designs for two-step protocols. *Australasian Journal of Combinatorics*, 2005. To appear.

[2] A. Baltag and L.S. Moss. Logics for epistemic programs. *Synthese*, 139:165–224, 2004. Knowledge, Rationality & Action 1–60.

[3] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8:18–36, 1990.

[4] K. Engelhardt, R. van der Meyden, and Y. Moses. Knowledge and the logic of local propositions. In I. Gilboa, editor, *Proceedings of TARK VII*, pages 29–41. Morgan Kaufmann, 1998.

[5] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge MA, 1995.

[6] M.J. Fischer and R.N. Wright. Bounds on secret key exchange using a random deal of cards. *Journal of Cryptology*, 9(2):71–99, 1996.

[7] P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In R. Alur and D. Peled, editors, *Proceedings of the 16th International conference on Computer Aided Verification (CAV 2004)*, pages 479–483. Springer, 2004.

[8] J.D. Gerbrandy and W. Groeneveld. Reasoning about information change. *Journal of Logic, Language, and Information*, 6:147–169, 1997.

[9] J.Y. Halpern, R. van der Meyden, and M.Y. Vardi. Complete axiomatizations for reasoning about knowledge and time. *SIAM Journal on Computing*, 33(3):674–703, 2004.

[10] K. Koizumi, T. Mizuki, and T. Nishizeki. Necessary and sufficient numbers of cards for the transformation protocol. In K.-Y. Chwa and J. Ian Munro, editors, *Computing and Combinatorics, 10th Annual International Conference (COCOON 2004)*, LNCS 3106, pages 92–101. Springer, 2004.

[11] Franco Raimondi and Alessio Lomuscio. Verification of multiagent systems via ordered binary decision diagrams: An algorithm and its implementation. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 630–637. IEEE Computer Society, 2004.

[12] R. Ramanujam and S. P. Suresh. Information based reasoning about security protocols. *Electr. Notes Theor. Comput. Sci.*, 55(1), 2001.

[13] W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In D. Bošnački and S. Leue, editors, *Model Checking Software, Proceedings of SPIN 2002 (LNCS Volume 2318)*, pages 95–111. Springer, 2002.

[14] R. van der Meyden. Common knowledge and update in finite environments. *Information and Computation*, 140(2):115–157, 1998.

[15] H.P. van Ditmarsch. Descriptions of game actions. *Journal of Logic, Language and Information*, 11:349–365, 2002.

[16] H.P. van Ditmarsch. The russian cards problem. *Studia Logica*, 75:31–62, 2003.

[17] H.P. van Ditmarsch. The case of the hidden hand. In *Liber Amicorum Dick de Jongh*, 2004. (electronically published) ISBN 90 5776 1289.

[18] J. van Eijck. Dynamic epistemic modelling. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam, 2004. CWI Report SEN-E0424.

[19] J. van Eijck and J. Ruan. Action emulation. manuscript, 2005.

[20] S. van Otterloo, W. van der Hoek, and M. Wooldridge. Model checking a knowledge exchange scenario. *Applied Artificial Intelligence*, 18(9-10):937–952, 2004.

REPORT OF INVENTIONS AND SUBCONTRACTS

(Pursuant to "Patent Rights" Contract Clause) (See Instructions on back)

Form Approved
OMB No. 9000-0095
Expires Oct 31, 2004

PLEASE DO NOT RETURN YOUR COMPLETED FORM TO THIS ADDRESS. RETURN COMPLETED FORM TO THE CONTRACTING OFFICER.

1.a. NAME OF CONTRACTOR/SUBCONTRACTOR: UNIVERSITY OF OTAGO

c. CONTRACT NUMBER: AGARD 054017

b. ADDRESS *(Include ZIP Code)*: RESEARCH & ENTERPRISE DUNEDIN 9001 NEW ZEALAND

d. AWARD DATE *(YYYYMMDD)*: 20040929

2.a. NAME OF GOVERNMENT PRIME CONTRACTOR:

c. CONTRACT NUMBER:

b. ADDRESS *(Include ZIP Code)*:

d. AWARD DATE *(YYYYMMDD)*:

3. TYPE OF REPORT *(X one)*: a. INTERIM [ ] b. FINAL [X]

4. REPORTING PERIOD *(YYYYMMDD)*
a. FROM 20041001
b. TO 20051001

SECTION I - SUBJECT INVENTIONS

5. "SUBJECT INVENTIONS" REQUIRED TO BE REPORTED BY CONTRACTOR/SUBCONTRACTOR *(If "None," so state)*

NAME(S) OF INVENTOR(S) *(Last, First, Middle Initial)* a.	TITLE OF INVENTION(S) b.	DISCLOSURE NUMBER, PATENT APPLICATION SERIAL NUMBER OR PATENT NUMBER c.	ELECTION TO FILE PATENT APPLICATIONS *(X)* d. (1) UNITED STATES (a) YES	(1) UNITED STATES (b) NO	(2) FOREIGN (a) YES	(2) FOREIGN (b) NO	CONFIRMATORY INSTRUMENT OR ASSIGNMENT FORWARDED TO CONTRACTING OFFICER *(X)* e. (a) YES	(b) NO
	NONE							

f. EMPLOYER OF INVENTOR(S) NOT EMPLOYED BY CONTRACTOR/SUBCONTRACTOR

(1) (a) NAME OF INVENTOR *(Last, First, Middle Initial)*

(b) NAME OF EMPLOYER

(c) ADDRESS OF EMPLOYER *(Include ZIP Code)*

(2) (a) NAME OF INVENTOR *(Last, First, Middle Initial)*

(b) NAME OF EMPLOYER

(c) ADDRESS OF EMPLOYER *(Include ZIP Code)*

g. ELECTED FOREIGN COUNTRIES IN WHICH A PATENT APPLICATION WILL BE FILED

(1) TITLE OF INVENTION

(2) FOREIGN COUNTRIES OF PATENT APPLICATION

SECTION II - SUBCONTRACTS *(Containing a "Patent Rights" clause)*

6. SUBCONTRACTS AWARDED BY CONTRACTOR/SUBCONTRACTOR *(If "None," so state)*

NAME OF SUBCONTRACTOR(S) a.	ADDRESS *(Include ZIP Code)* b.	SUBCONTRACT NUMBER(S) c.	FAR "PATENT RIGHTS" d. (1) CLAUSE NUMBER	(2) DATE *(YYYYMM)*	DESCRIPTION OF WORK TO BE PERFORMED UNDER SUBCONTRACT(S) e.	SUBCONTRACT DATES *(YYYYMMDD)* f. (1) AWARD	(2) ESTIMATED COMPLETION
		NONE					

SECTION III - CERTIFICATION

7. CERTIFICATION OF REPORT BY CONTRACTOR/SUBCONTRACTOR *(Not required if: (X as appropriate))* [ ] SMALL BUSINESS or [ ] NONPROFIT ORGANIZATION

I certify that the reporting party has procedures for prompt identification and timely disclosure of "Subject Inventions," that such procedures have been followed and that all "Subject Inventions" have been reported.

a. NAME OF AUTHORIZED CONTRACTOR/SUBCONTRACTOR OFFICIAL *(Last, First, Middle Initial)*: MRS JO-ANNE SKINNER

b. TITLE:

c. SIGNATURE: JS

d. DATE SIGNED: 23/2/06

 PREVIOUS EDITION IS OBSOLETE.

Department of Computer Science,
University of Otago

Technical Report OUCS-2006-01

Sum and Product in Dynamic Epistemic Logic

Authors:

H. P. van Ditmarsch
Computer Science, University of Otago

J. Ruan
Computer Science, University of Liverpool, United Kingdom

L.C. Verbrugge
Artificial Intelligence, University of Groningen, Netherlands

Status: Journal submission

Department of Computer Science,
University of Otago, PO Box 56, Dunedin, Otago, New Zealand

http://www.cs.otago.ac.nz/research/techreports.html

Sum and Product in Dynamic Epistemic Logic*

H.P. van Ditmarsch[†], J. Ruan[‡], and L.C. Verbrugge[§]

Abstract

The Sum-and-Product riddle was first published in [Fre69]. We provide an overview on the history of the dissemination of this riddle through the academic and puzzle-math community. This includes some references to precursors of the riddle, that were previously (as far as we know) unknown.

We then model the Sum-and-Product riddle in a modal logic called public announcement logic. This logic contains operators for knowledge, but also operators for the informational consequences of public announcements. The logic is interpreted on multi-agent Kripke models. The information in the riddle can be represented in the traditional way by number pairs, so that Sum knows their sum and Product their product, but also as an interpreted system, so that Sum and Product at least know their local state. We show that the different representations are isomorphic. We also provide characteristic formulas of the initial epistemic state of the riddle. Finally we analyze one of the announcements towards the solution of the riddle as a so-called unsuccessful update: a formula that become false because it is announced.

The riddle is then implemented and its solution verified in the epistemic model checker DEMO. This can be done, we think, surprisingly elegantly. The results are compared with other work in epistemic model checking.

Keywords: modal logic, puzzle math, dynamic epistemics, characteristic formula, model checking

1 Introduction

The following problem, or riddle, was first stated, in the Dutch language, in [Fre69] and subsequently solved in [Fre70]. A translation of the original formulation is:

*Contact author is Hans van Ditmarsch. Hans and Ji appreciate support from AOARD research grant AOARD-05-4017. Hans and Rineke appreciate support from the Netherlands Organization for Scientific Research (NWO). A shortened version of Sections 7 and 8, including Figure 2, have previously appeared in conference proceedings [vDRV05].

[†]Computer Science, University of Otago, New Zealand, hans@cs.otago.ac.nz

[‡]Computer Science, University of Liverpool, United Kingdom, jruan@csc.liv.ac.uk

[§]Artificial Intelligence, University of Groningen, the Netherlands, rineke@ai.rug.nl

No. 223. A zegt tot S en P: Ik heb twee gehele getallen x, y gekozen met $1 < x < y$ en $x + y \leqslant 100$. Straks deel ik $s = x + y$ aan S alleen mee, en $p = xy$ aan P alleen. Deze mededelingen blijven geheim. Maar jullie moeten je inspannen om het paar (x, y) uit te rekenen.

Hij doet zoals aangekondigd. Nu volgt dit gesprek:

1. P zegt: Ik weet het niet.
2. S zegt: Dat wist ik al.
3. P zegt: Nu weet ik het.
4. S zegt: Nu weet ik het ook.

Bepaal het paar (x, y).

(*H. Freudenthal*).

Figure 1: The original publication

> A says to S and P: I have chosen two integers x, y such that $1 < x < y$ and $x + y \leq 100$. In a moment, I will inform S only of $s = x + y$, and P only of $p = xy$. These announcements remain private. You are required to determine the pair (x, y).
>
> He acts as said. The following conversation now takes place:
>
> i. P says: “I do not know it.”
>
> ii. S says: “I knew you didn’t.”
>
> iii. P says: “I now know it.”
>
> iv. S says: “I now also know it.”
>
> Determine the pair (x, y).

This problem is considered a riddle, or puzzle, because the agents’ announcements *appear* to be uninformative, as they are about ignorance and knowledge and not about (numerical) facts, whereas *actually* they are very informative: the agents learn facts from the other’s announcements. For example, the numbers cannot be 2 and 3, or any other pair of prime numbers, nor for example 2 and 4, because in all those cases Product would immediately have deduced the pair from their product. As a somewhat more complicated example, the numbers cannot be 14 and 16: if they were, their sum would be 30. This is also the sum of the prime numbers 7 and 23. But then, as in the previous example, Product would (P) would have known the numbers, and therefore Sum (S) – if the sum had been 30 – would have considered it possible that Product knew the numbers. But Sum said that he *knew* that Product didn’t know the numbers. So the numbers cannot be 14 and 16. Sum and Product learn enough, by eliminations

of which we gave some examples, to be able to determine the pair of numbers: the unique solution of the problem is the pair $(4, 13)$.

The knowledge that agents have about other agents' mental states and, in particular, about the effect of communications, is vital for solving important problems in multi-agent systems, both for cooperative and for competitive groups. Dynamic epistemic logic was developed to study the changes brought about by communication in such higher-order knowledge of other agent's and of group knowledge [BMS98, Ger99]. The Sum-and-Product puzzle presents a complex illustrative case of the strength of specifications in dynamic epistemic logic and of the possibilities of automated model checking, and both can also be used in real multi-agent system applications. As far as we know, we are the first to use an automated model checker to tackle the Sum-and-Product problem.

Section 2 gives an overview of the dissemination of the riddle through the academic community, and suggests some precursors. In Section 3 we introduce public announcement logic. In Section 4 we model the Sum-and-Product problem in public announcement logic. Section 5 models the Sum-and-Product problem, alternatively, as an interpreted system, and Section 6 provides the general setting of unsuccessful updates of which some announcements in the riddle provide examples. In Section 7 we introduce the epistemic model checker DEMO. In Section 8 we implement the Sum-and-Product specification of Section 4 in DEMO, and we verify its epistemic features. Section 9 reports on DEMO implementations of similar problems, and compares the model checking results to our experiences with other epistemic model checkers.

2 History

John McCarthy wrote the earliest full-length treatment of the Sum-and-Product riddle in the years 1978–1981 [McC90]. McCarthy formulates the problem as follows:

> Two numbers m and n are chosen such that $2 \leq m \leq n \leq 99$. Mr. S is told their sum and Mr. P is told their product. The following dialogue ensues:
>
> i. Mr. P: I don't know the numbers.
>
> ii. Mr. S: I knew you didn't know. I don't know either.
>
> iii. Mr. P: Now I know the numbers.
>
> iv. Mr. S: Now I know them too.
>
> In view of the above dialogue, what are the numbers?

In [McC90] the problem is elegantly modeled in modal logic in such a way that it can be processed in the (first-order) logic theorem prover FOL. This includes an – almost off-hand – introduction of what corresponds to the essential concept

of ‘common knowledge’: what Sum and Product commonly know is crucial to a clear understanding of the problem. Common knowledge had received a very interesting treatment already in Lewis’ 1969 book *Convention* [Lew69] (and also appears in other philosophical literature from that period, e.g. in Schiffer’s work [Sch72]), but McCarthy seems to have re-invented it in his 1981 article, thereby inspiring research on common knowledge in Artificial Intelligence.

Note that in the second announcement, Sum seems to give some additional information that does not appear in the Freudenthal-version of the dialogue, namely “I don’t know either”. However, some simple considerations show that this addition is superfluous, because at the current point in the dialogue, it is already common knowledge among the participants that Sum doesn’t know either. After all, the only situation in which Sum *does* know the two numbers from the start is the one where the pair of numbers is $(2, 3)$, which has already been ruled out by Product’s first announcement. Further, note that McCarthy allows the two numbers to be the same, unlike Freudenthal. This also does not affect the solution.

Many different versions of the puzzle elicited much discussion from the late seventies onwards. The variations are caused by different announcements, different ranges for the numbers, and different choices for what is considered to be common knowledge at the starting-point. For yet another example, for a certain larger range of possible numbers than 2—99 one finds a solution different from $(4, 13)$ but that then after all is in the 2—99 range. Discussions of several variants of the problem can be found in the literature on recreational mathematics, see especially [Gar79, Sal95, Isa95], and a website `www.mathematik.uni-bielefeld.de/~sillke/PUZZLES/logic_sum_product` that contains many other references.

More geared towards an epistemic logical audience are [Pla89, Pan91, vdM94, vdHV02]. Plaza and Panti were students of Rohit Parikh and have both made some interesting contributions to epistemic logic. In [Pla89] the Sum-and-Product problem is modeled in a dynamic epistemic logic that is the precursor of the public announcement logic presented here, namely without an operator for common knowledge. In [Pan91], on the other hand, the common knowledge involved in the Sum-and-Product puzzle is investigated in detail, with an emphasis on the arithmetic involved. For example, for the formulation of the problem where the range of numbers (up to 100) is not considered to be common knowledge at the start, Panti proves that if the sum of the numbers is greater or equal than 7, then this (and its logical consequences) is the *only* fact that is common knowledge among Sum and Product. Finally, Van der Meyden [vdM94] suggests a solution in temporal epistemic logic.

2.1 Looking for the origin of Sum and Product

In both of the two first full-length publications on the Sum and Product riddle [McC90, Gar79], the authors explicitly wondered about but could not give its exact origins. John McCarthy explains in a footnote in his paper [McC90]:

> I have not been able to trace Mr. S and Mr. P back beyond its alleged appearance on a bulletin board at Xerox PARC.

Martin Gardner, in his 1979 "Mathematical Games" column [Gar79], writes:

> This beautiful problem, which I call "impossible" because it seems to lack sufficient information for a solution, began making the rounds of mathematics meetings a year or so ago. I do not know its origin.

After the appearance of [Gar79], the fact that the puzzle had been published already in 1969 by Dutch topologist and specialist on mathematics education Hans Freudenthal, was brought to Gardner's attention by Dutch algebraist Robert van der Waall. Van der Waall was one of the small number of Dutch mathematicians who had sent in a correct solution to the Dutch mathematics journal *Nieuw Archief voor Wiskunde* after the puzzle's first appearance in 1969.

We have tried to fill in two missing pieces in the history of the Sum-and-Product riddle:

i. If [Fre69] is indeed the first published appearance of the problem, then how did the problem migrate from the Dutch mathematics community of the late 1960s and early 1970s to "a bulletin board at Xerox Parc" and "the rounds of mathematics meetings" in the United States in the late 1970s?

ii. Did Freudenthal invent the problem? And if so, has he possibly been inspired by (less complex) precursors?

Despite several requests on international e-mail lists, we have not been able to answer the first question. As to the second question, we received a partial answer from one of the subscribers to *Nieuw Archief voor Wiskunde*, who thought he remembered to have seen the Sum-and-Product riddle in the puzzle column "Breinbrouwsels" (brain brews) in the now defunct Dutch-language weekly *De Katholieke Illustratie* ('Illustrated Catholic Magazine') in the 1950s.

We have visited several libraries and thus managed to read almost all of the 626 "Breinbrouwsels" that G. van Tilburg published from 1954 until 1965 (and of those we did not read, we could infer what they were from their answers, in other issues). This did not turn up the Sum-and-Product puzzle, but we did find four puzzles (published in 1954, 1955, 1957, and 1963, respectively) that can clearly be seen as precursors. Mostly these puzzles involve partial information about a number of persons' ages, where the fact that one of the participants cannot deduce the ages from the interlocutor's hint, but can deduce them after some further dialogue, is crucial information helping the reader to solve the problem. We will describe some of Van Tilburg's interesting problems in a forthcoming publication in Dutch.

Thus, as far as we know now, Freudenthal really invented the Sum-and-Product puzzle, but may have been inspired by Van Tilburg's "Breinbrouwsels". Possibly he also read some even earlier riddles of British origin, to which we turn our attention now.

2.2 Precursors of Sum and Product

David Singmaster's bibliographies on recreational mathematics (see e.g. `www.g4g4.com/MyCD5/SOURCES/singmaterial.htm`) point to some candidate epistemic puzzles that appeared even earlier than Van Tilburg's. The earliest precursor of the Sum-and-Product riddle that we have been able to trace is the following one, probably invented by Williams and Savage and first published in book-form in 1940 in *The Penguin Problems Book* [WS40, p.53]:

> **The church afloat**
> "I'm taking three females on the river to-morrow," said the vicar to his curate; "would you care to join our party?"
>
> "What are their ages?" asked the curate, cautiously. "Far be it from me to disclose a lady's age!" said the vicar, "but I can tell you this – the product of their ages is 840, and the sum is twice the number of years in your own age. You, a mathematician, should be able to find their ages for yourself."
>
> "Sounds like casuistry, Vicar," said the curate; "but, as a matter of fact, I can't find their ages from your data. By the way, is the eldest older than you?"
>
> "No, younger." "Ah, now I know their ages!" said the curate. "Thanks, I will come with pleasure."
>
> What was the curate's age? How old were the ladies? And what can be deduced about the vicar's age?

Here follows Williams' and Savage's answer [WS40, p.135]:

> Sum of ages must be even.
>
> Uncertainty, resolved by the vicar's final statement, must be due to the fact of there being more than one such sum which was twice the curate's age.
>
> Of the possible sets of 3 factors of 840, there are only two cases of the same *even* sum occurring more than once. The sums in these cases are 46 and 30. Now the curate's age could not be 15; therefore he was 23.
>
> The sets of female ages giving a sum of 46 are 35, 8, 3 and 30, 14, 2. Since the vicar's answer excluded one of these, that one must be the former. Therefore the ladies' ages were 30, 14, 2, and the vicar's age must lie between 30 and 35.

Note that some world knowledge is used implicitly here, namely the fact that mathematicians (and curates) are always older than 15 years, and the fact that the curate, being a mathematician, reasons correctly.

Another problem, that was published in 1944 in *The Second Penguin Problems Book* [WS44, p.27], also hinges on the fact that only for some number

combinations there is more than one way to make the same sum. In a way, the next problem is less attractive than the previous one, because the uncertainty is not completely dissolved at the end: readers are asked to derive the sum of the ages only.

> **Domiciliary**
> "I have told you my age," said Mr. Ptolemy to the inspector who had just knocked on his door. "Besides myself, there are three persons living in this house; the product of their ages is one thousand two hundred ninety-six, and the sum of their ages is the number of the house."
>
> "But it is impossible for me to be *sure* of their ages without further information," said the inspector. "Is any one of them the same age as yourself?"
>
> "No," said Mr. Ptolemy.
>
> "Thanks; now I know their ages," said the inspector.
>
> What was the number of Mr. Ptolemy's house?

This time, the explanation is as follows [WS44, p.116]; again, the authors implicitly use some world knowledge:

> There are many ways of splitting 1296 into three factors, but only *possible* ones need be considered. Two of these sets of factors have the same sum, namely 1, 18, 72 and 2, 8, 81, adding up to 91. The other sums are all different.
>
> As the inspector could not be sure of the ages from the fact that they added up to the number of the house (which he, of course, knew), this number must have been 91.
>
> [Mr. Ptolemy's age - also known to the inspector - must have been 72 or 81 (unless it was 18 or 8 - both unlikely), but we have no means of deciding this point.]

The above two puzzles are roughly of the same kind as Van Tilburg's, but still different. In fact, Van Tilburg may have been inspired to create his puzzles after reading the British gentlemen. Essentially the same problem as "Domiciliary", but in a somewhat different guise, was printed in Greenblatt's *Mathematical Entertainments* [Gre68], first published in the United States in 1965. Greenblatt starts with some historical speculation:

> One of the few amusing things to come out of World War II was a new type of brain twister - the "census-taker" problem. (The time and place of origin of a problem are difficult to specify. To the best of the author's knowledge, this problem was born on the M.I.T. campus in one of the war projects.)

As we now know, the type of problem probably stems from at least somewhat before the start of World War II, and from Great Britain instead of the United States. After all, *The Penguin Problems Book*, although published during the War in 1940, was mostly based on earlier puzzles from Williams' and Savage's column "Perplexities" that used to appear in *The Strand Magazine*. For more details, see `www.cs.otago.ac.nz/staffpriv/hans/sumpro/`.

2.3 Descendants of Sum and Product

In recent years, variants of the Sum-and-Product riddle keep cropping up. Johan van Benthem has communicated a particularly nice example, dubbed "GSM-puzzle". The conversation between the two participants in the GSM-puzzle follows exactly the same pattern as the one in the Sum-and-Product riddle. However, due to the context in terms of playing cards with points and colors, no arithmetic is needed to solve it. Thus, the epistemic complexity remains, while the arithmetic complexity has been canceled. For a formulation of the problem, see `www.ai.rug.nl/mas/openprojecten.html\#GSM` and/or `www.cs.otago.ac.nz/staffpriv/hans/sumpro/`.

Some of the more recent variants include more than two participants in the clarifying conversation, for example the following one [Liu04], which combines themes from the Muddy-Children puzzle [MDH86] with those from the Sum-and-Product puzzle. We leave this problem as a challenge to the reader.

> Each of Ace, Bea and Cec is wearing a hat on which a positive integer is printed. Each can see only the numbers on the others' hats. They are told that one of the numbers is the sum of the other two. They make the following statements in succession.
>
> i. Ace: I cannot deduce what my number is.
> ii. Bea: Knowing that, I still cannot deduce what my number is.
> iii. Cec: Knowing that, I still cannot deduce what my number is.
> iv. Ace: Now I can deduce that my number is 50.
>
> Assuming that they all use sound reasoning, what are the numbers on the two other hats?

After this detailed overview of the dissemination of the Sum-and-Product riddle, which we hope may prevent some of this information from gradually disappearing into the fog of war on academic battlegrounds, we continue with the more technical core of this paper, that consists of an introduction into public announcement logic, modelling the riddle in this logic, and verifying its properties in a model checker.

3 Public Announcement Logic

Public announcement logic is a dynamic epistemic logic and is an extension of standard multi-agent epistemic logic. Intuitive explanations of the epistemic part of the semantics can be found in [FHMV95, vdHV02, vDvdHK05]. We give a concise overview of, in that order, the language, the structures on which the language is interpreted, and the semantics.

Given are a finite set of agents N and a finite or countably infinite set of atoms Q. The language of public announcement logic is inductively defined as

$$\varphi ::= q \mid \neg\varphi \mid (\varphi \wedge \psi) \mid K_n\varphi \mid C_G\varphi \mid [\varphi]\psi$$

where $q \in Q$, $n \in N$, and $G \subseteq N$ are arbitrary. For $K_n\varphi$, read 'agent n knows formula φ'. For $C_G\varphi$, read 'group of agents G commonly know formula φ'. For $[\varphi]\psi$, read 'after public announcement of φ, formula ψ (is true)'.

Next, we introduce the structures. An *epistemic model* $M = \langle W, \sim, V\rangle$ consists of a *domain* W of (factual) *states* (or 'worlds'), *accessibility* $\sim : N \rightarrow \mathcal{P}(W \times W)$, where each $\sim(n)$ is an equivalence relation, and a *valuation* $V : Q \rightarrow \mathcal{P}(W)$. For $w \in W$, (M, w) is an *epistemic state* (also known as a pointed Kripke model). For $\sim(n)$ we write $\sim_n$, and for $V(q)$ we write V_q. So, accessibility $\sim$ can be seen as a set of equivalence relations $\sim_n$, and V as a set of valuations V_q. Given two states w, w' in the domain, $w \sim_n w'$ means that w is indistinguishable from w' for agent n on the basis of its information. For example, at the beginning of the riddle, pairs $(14, 16)$ and $(7, 23)$ are indistinguishable for Sum but not for Product. Therefore, assuming a domain of number pairs, we have that $(14, 16) \sim_S (7, 23)$ but that $(14, 16) \not\sim_P (7, 23)$. The group accessibility relation $\sim_G$ is the transitive and reflexive closure of the union of all accessibility relations for the individuals in G: $\sim_G \equiv (\bigcup_{n \in G} \sim_n)^*$. This relation is used to interpret common knowledge for group G.

Finally, we give the semantics. Assume an epistemic model $M = \langle W, \sim, V\rangle$.

$$\begin{array}{lll}
M, w \models q & \text{iff} & w \in V_q \\
M, w \models \neg\varphi & \text{iff} & M, w \not\models \varphi \\
M, w \models \varphi \wedge \psi & \text{iff} & M, w \models \varphi \text{ and } M, w \models \psi \\
M, w \models K_n\varphi & \text{iff} & \text{for all } \ v \in W : w \sim_n v \text{ implies } M, v \models \varphi \\
M, w \models C_G\varphi & \text{iff} & \text{for all } \ v \in W : w \sim_G v \text{ implies } M, v \models \varphi \\
M, w \models [\varphi]\psi & \text{iff} & M, w \models \varphi \text{ implies } M|\varphi, w \models \psi
\end{array}$$

Here, epistemic model $M|\varphi = \langle W', \sim', V'\rangle$ is defined as

$$\begin{array}{lll}
W' & = & \{w' \in W \mid M, w' \models \varphi\} \\
\sim'_n & = & \sim_n \cap (W' \times W') \\
V'_q & = & V_q \cap W'
\end{array}$$

The dynamic modal operator $[\varphi]$ is interpreted as an epistemic state transformer. Announcements are assumed to be truthful, and this is commonly known by all agents. Therefore, the model $M|\varphi$ is the model M restricted to all the

states where φ is true, including access between states. The dual of $[\varphi]$ is $\langle\varphi\rangle$: $M, w \models \langle\varphi\rangle\psi$ iff $M, w \models \varphi$ and $M|\varphi, w \models \psi$.

Formula φ is valid on model M, notation $M \models \varphi$, if and only if for all states w in the domain of M: $M, w \models \varphi$. Formula φ is valid, notation $\models \varphi$, if and only if for all models M: $M \models \varphi$. Logical consequence $\Psi \models \varphi$ is defined as "for all (M, w), if $M, w \models \psi$ for all $\psi \in \Psi$, then $M, w \models \varphi$." For $\{\psi\} \models \varphi$, write $\psi \models \varphi$.

A proof system for this logic is presented, and shown to be complete, in [BMS98], with precursors – namely for public announcement logic *without* common knowledge – in [Pla89, Ger99]. For a concise completeness proof, see [vDvdHK05]. Some relevant principles of this logic are

i. $[\varphi]\psi \leftrightarrow (\varphi \rightarrow [\varphi]\psi)$

ii. $[\varphi][\psi]\chi \leftrightarrow [\varphi \wedge [\varphi]\psi]\chi$

iii. $[\varphi]K_n\psi \leftrightarrow (\varphi \rightarrow K_n[\varphi]\psi)$

iv. $[C_N\varphi]C_N\varphi$

Item i expresses that the interpretation of the dynamic operator $[\varphi]$ is a *partial* function. Item ii expresses that a sequence of two announcements φ and ψ can be replaced by the single announcement 'φ, and after φ, ψ'. Item iii expresses the preconditions and postconditions of announcements with respect to individual knowledge (for common knowledge, this relation is more complex). Item iv expresses that *public* knowledge (i.e., common knowledge for the entire group of agents) remains true after announcement. Not all formulas remain true after their announcement, in other words, $[\varphi]\varphi$ is *not* a principle of this logic. This matter will be addressed in Section 6. Some announcements towards the solution of the Sum-and-Product problem provide concrete counterexamples, and this will explain why the 'puzzling' conversation of S and P makes sense.

4 Sum and Product in Public Announcement Logic

We give a specification of the Sum-and-Product problem in public announcement logic. First we need to determine the set of atomic propositions and the set of agents. In the formulation of the problem, x, y are two integers such that $1 < x < y$ and $x+y \leq 100$. Define $I \equiv \{(x, y) \in \mathbb{N}^2 \mid 1 < x < y \text{ and } x+y \leq 100\}$. Consider the variable x. If its value is 3, we can represent this information as the (truth of) the atomic proposition '$x = 3$'. Slightly more formally we can think of '$x = 3$' as a propositional letter x_3. Thus we create a (finite) set of atoms $\{x_i \mid (i, j) \in I\} \cup \{y_j \mid (i, j) \in I\}$.

Concerning the agents, the role of the announcer A is to guarantee that the background knowledge for solving the problem is commonly known among Sum and Product. The announcer need not be introduced as an agent in the logical

modelling of the system. That leaves $\{S, P\}$ as the set of agents. Agents S and P will also be referred to as Sum and Product, respectively.

The proposition 'Sum knows that the numbers are 4 and 13' is represented as $K_S(x_4 \wedge y_{13})$. The proposition 'Sum knows the (pair of) numbers' is described as $K_S(x, y) \equiv \bigvee_{(i,j)\in I} K_S(x_i \wedge y_j)$. Similarly, 'Product knows the numbers' is represented by $K_P(x, y) \equiv \bigvee_{(i,j)\in I} K_P(x_i \wedge y_j)$. Furthermore, note that the 'knew' in announcement ii, by Sum, refers to the truth of $K_S \neg K_P(x, y)$ in the *initial* epistemic state, not in the epistemic state *resulting* from announcement i, by Product. Therefore, announcement i by Product is superfluous in the subsequent analysis.[1] This is sufficient to formalize the announcements made towards a solution of the problem:

i. P says: "I do not know it": $\neg K_P(x, y)$

ii. S says: "I knew you didn't": $K_S \neg K_P(x, y)$

iii. P says: "I now know it": $K_P(x, y)$

iv. S says: "I now also know it": $K_S(x, y)$

We can interpret these statements on an epistemic model $\mathcal{SP}_{(x,y)} \equiv \langle I, \sim, V\rangle$ consisting of a domain of all pairs $(x, y) \in I$ (as above), with accessibility relations $\sim_S$ and $\sim_P$ such that for Sum: $(x, y) \sim_S (x', y')$ iff $x + y = x' + y'$, and for Product: $(x, y) \sim_P (x', y')$ iff $xy = x'y'$; and with valuation V such that $V_{x_i} = \{(x, y) \in I \mid x = i\}$ and $V_{y_j} = \{(x, y) \in I \mid y = j\}$.

We can describe the solution of the problem as the truth of the statement

$$\mathcal{SP}_{(x,y)}, (4, 13) \models \langle K_S \neg K_P(x, y)\rangle\langle K_P(x, y)\rangle\langle K_S(x, y)\rangle\top$$

This expresses that, if $(4, 13)$ is the initial state, then it is possible to publicly announce *ii*, *iii*, and *iv*, in that order. We can also express more properly that $(4, 13)$ is the only solution as the model validity

$$\mathcal{SP}_{(x,y)} \models [K_S \neg K_P(x, y)][K_P(x, y)][K_S(x, y)](x_4 \wedge y_{13})$$

5 Sum and Product as an interpreted system

A relevant observation is that a pair of numbers (x, y) with $x < y$ corresponds to exactly one sum-product pair (s, p). In one direction this is trivial, for the other direction: assume $(x+y, xy) = (x'+y', x'y')$. Let without loss of generality x be the smaller of x and x', so that $x' = x+v$. Then from $xy = x'y' = (x+v)(y-v)$ follows that $yv - xv - v^2 = 0$, so that $v = 0$ or $v = y - x$. The second merely reverses the role of x and y; in our terms, it cannot be satisfied, because x was required to be strictly smaller than y. This observation paves the way

[1] In dynamic epistemic logic *with assignment* one can model such past tense epistemic statements explicitly [Koo05].

for a different modelling of the problem than the traditional one with '(smaller number, larger number)' pairs (x, y).

We now let atomic propositions represent the sum and product of the different numbers, instead of representing these numbers themselves. For example, s_7 represents that the sum of the two numbers is 7. We allow a slight abuse of the language: if $i+j = k$ then we also write s_{i+j} for s_k. Similarly, we write p_{ij} for p_l when $ij = l$. Thus we create a set of atoms $\{s_{x+y} \mid (x, y) \in I\} \cup \{p_{xy} \mid (x, y) \in I\}$.

The obvious way to interpret *such* atoms is on an epistemic model $\mathcal{SP}_{(s,p)} \equiv \langle W', \sim', V' \rangle$ with a domain W' consisting of all pairs (s, p) such that $s = x + y$ and $p = xy$ (as in the original formulation of the problem) for all $(x, y) \in I$, i.e., with $1 < x < y$ and $x + y \leq 100$; with *accessibility relations* $\sim'_S$ and $\sim'_P$ such that for Sum: $(s, p) \sim'_S (s', p')$ iff $s = s'$, and for Product: $(s, p) \sim'_P (s', p')$ iff $p = p'$; and with valuation such that $V'_{s_{x+y}} = \{(s, p) \in W' \mid s = x + y\}$ and $V'_{p_{xy}} = \{(s, p) \in W' \mid p = xy\}$.

We have now modelled the problem as an *interpreted system* where agents at least know their local state. Interpreted systems were introduced in theoretical computer science as an abstract architecture for distributed systems [FHMV95]. Sum's local state is the sum of the two numbers, Product's local state is the product of the two numbers. A global state for the problem is a pair of local states, one for Sum and one for Product. The set of global states is a subset of the full cartesian product of local state values: the dependencies between local states enable Sum and Product to communicate their local state to each other without explicitly referring to it.

'Sum knows the (pair of) numbers' can be represented by 'Sum knows the global state of the system', i.e., as $K_S(s, p) \equiv \bigvee_{(x,y)\in I} K_S(s_{x+y} \wedge p_{xy})$, and, similarly, 'Product knows the numbers' by $K_P(s, p) \equiv \bigvee_{(x,y)\in I} K_P(s_{x+y} \wedge p_{xy})$. The formalization of the announcements made towards a solution of the problem is then similar to above:

$$\mathcal{SP}_{(s,p)} \models [K_S \neg K_P(s, p)][K_P(s, p)][K_S(s, p)](s_{4+13} \wedge p_{4\cdot 13})$$

An advantage of this representation is that we can apply known results for interpreted systems, such that agents at least know their local state, and the availability of *characteristic formulas* for modal structures [BM96, vB98] to the specific case of finite interpreted systems [vDvdHK03]. That agent S knows its local state, means that S knows the sum of the two numbers, whatever they are: $\mathcal{SP}_{(s,p)} \models s_{x+y} \to K_S s_{x+y}$. From this follows that in the models for our problem a requirement $K_S(s_{x+y} \wedge p_{xy})$, that is equivalent to $K_S s_{x+y} \wedge K_S p_{xy}$, is equivalent to $K_S p_{xy}$. Similarly, $p_{xy} \to K_P p_{xy}$, and therefore, in the models, $K_P(s_{x+y} \wedge p_{xy})$ is equivalent to $K_P s_{x+y}$.

Concerning the characteristic formula describing the initial situation, we can apply results from [vDvdHK03].[2] The characteristic formula $\delta(\mathcal{SP}_{(s,p)})$ is

[2] A characteristic formula of a pointed model (M, w) is a formula $\delta(M, w)$ such that $M, w \models \psi$ iff $\delta(M, w) \models \psi$, in other words, any ψ true in (M, w) is entailed by $\delta(M, w)$. A similar notion equates model validity with entailment by way of $M \models \psi$ iff $\delta(M) \models \psi$. These descriptions exist for finite epistemic models. We also have that $\delta(M, w) \leftrightarrow (\delta(w) \wedge C_N \delta(M))$,

defined as

$$\begin{array}{rcl}\delta(\mathcal{SP}_{(s,p)}) & \equiv & \bigvee_{(x,y)\in I}(s_{x+y} \wedge p_{xy}) \;\; \wedge \\ & & \bigwedge_{(x,y)\in I}(K_S s_{x+y} \leftrightarrow \neg K_S \neg(s_{x+y} \wedge p_{xy})) \;\; \wedge \\ & & \bigwedge_{(x,y)\in I}(K_P p_{xy} \leftrightarrow \neg K_P \neg(s_{x+y} \wedge p_{xy}))\end{array}$$

The first conjunct of $\delta(\mathcal{SP}_{(s,p)})$ sums up the valuations of the different states in the domain. The second conjunct says (entails) that S knows its local state if and only if it considers possible any global state with that local state. For example $K_S s_{17} \leftrightarrow \neg K_S \neg(s_{17} \wedge p_{52})$; another conjunct is $K_S s_{17} \leftrightarrow \neg K_S \neg(s_{17} \wedge p_{60})$. From this follows that $K_S s_{17}$ implies $\neg K_S \neg p_{52} \wedge \neg K_S \neg p_{60} \wedge \ldots$: if the sum of the two numbers is 17, S considers it possible that their product is 52, or 60, etc.

The traditional modelling of Sum and Product relates to the interpreted system modelling in a precise technical sense. Expand the language to one containing atoms for all numbers x, y *and* atoms for all sums and products s, p of those numbers. Extend the models $\mathcal{SP}_{(x,y)}$ and $\mathcal{SP}_{(s,p)}$ to $\mathcal{SP}^+_{(x,y)}$ and $\mathcal{SP}^+_{(s,p)}$, respectively, by adding valuations for all sum and product atoms in the former, and for all smaller and larger number atoms in the latter. For example, to define $\mathcal{SP}^+_{(x,y)}$ we have to add valuations for all atoms s and p such that $(x, y) \in V^+_{s_{x+y}}$ iff $s = x+y$ and $(x, y) \in V^+_{p_{xy}}$ iff $p = xy$. We now have that $\mathcal{SP}^+_{(x,y)}$ and $\mathcal{SP}^+_{(s,p)}$ are isomorphic. (From this then follows that the models are also *bisimilar* [BdRV01] – a slightly weaker notion of 'sameness of models' that still guarantees that the theories describing the models are logically equivalent.) Without going into great detail, it suffices to define the isomorhphism as $\mathfrak{R} : I \rightarrow W'$ such that $\mathfrak{R} : (x, y) \mapsto (x + y, xy)$, to observe that this relation is a bijection, that $(x, y) \sim_S (x', y')$ iff $\mathfrak{R}(x, y) \sim_S \mathfrak{R}(x', y')$ iff $(x + y, xy) \sim_S (x' + y', x'y')$, and similarly for Product, and that the valuation of all facts remains the same for any states (x, y) and $(x + y, xy)$. The characteristic formula for the interpreted system $\mathcal{SP}^+_{(s,p)}$ in the expanded logical language is the previous one, $\delta(\mathcal{SP}_{(s,p)})$, in conjunction with

$$\bigwedge_{(i,j)\in I}((x_i \wedge y_j) \leftrightarrow (s_{i+j} \wedge p_{ij}))$$

This *propositional* equivalence relates a number pair to its unique corresponding sum and product pair.

To conclude, using the interpreted system representation, we can describe the initial situation for the Sum-and-Product puzzle in a very precise way. Moreover, the traditional representation and the interpreted system one are in a sense interchangeable: they have the same logical theory.

---

where $\delta(w)$ is *the description of state* w, for example summing up its valuation, or some other formula only true in w.

6 Unsuccessful updates

Not all formulas remain true after their announcement, in other words, $[\varphi]\varphi$ is *not* a principle of public announcement logic. A poignant example is when I'm telling you that "You don't know that the Highlanders just beat the Lions!". In the standard conversational setting this presumes that the factual information of which you are ignorant is actually the case, i.e., this normally means "The Highlanders just beat the Lions and you don't know that the Highlanders just beat the Lions." It is therefore an announcement of the form $q \wedge \neg K_n q$. After the announcement, you know that the fact in question is true – $K_n q$ – and therefore the formula of the announcement has become false: $K_n q$ entails $\neg q \vee K_n q$, which is equivalent to $\neg(q \wedge \neg K_n q)$, the negation of the announcement. In a somewhat different setting that the formula $q \wedge \neg K_n q$ cannot be consistently known, this phenomenon has been known in philosophical circles for a long time, namely as the Moore-paradox [Moo42, Hin62]. In the underlying dynamic setting it has been described as an unsuccessful update in [Ger99, Ger05]. General terminology is proposed in [vDK05]. Let φ be a formula in the language of public announcement logic:

- *Successful formula*
  φ is successful iff $[\varphi]\varphi$ is valid.
- *Unsuccessful formula*
  φ is unsuccessful iff it is not successful.
- *Successful update*
  φ is successful in epistemic state (M, w) iff $M, w \models \langle\varphi\rangle\varphi$
- *Unsuccessful update*
  φ is unsuccessful in (M, w) iff $M, w \models \langle\varphi\rangle\neg\varphi$.

Note that an unsuccessful formula may be a successful update in one epistemic state and an unsuccessful update in another epistemic state. It can be shown that $[\varphi]\varphi$ is valid iff $[\varphi]C_G\varphi$ is valid iff $\varphi \to [\varphi]C_G\varphi$ is valid. (See [vDK05], the second equivalence follows directly from the principle $[\varphi]\psi \leftrightarrow (\varphi \to [\varphi]\psi)$, listed as item i on page 10 in Section 3.) Therefore, the successful formulas capture the notion 'formulas that remain true after their announcement'.

Clearly, also in the course of solving the Sum-and-Product problem the agents appear to learn things that they did not know before. So some reversal of ignorance into knowledge seems to take place. We therefore expect that some of the announcements made towards the solution of the problem are unsuccessful updates. In this section we refer to those four successive announcements as (how they have been enumerated before, namely as) (*i*) $\neg K_P(x, y)$, (*ii*) $K_S\neg K_P(x, y)$, (*iii*) $K_P(x, y)$, and (*iv*) $K_S(x, y)$.

The case i Remember that announcement i was superfluous in the analysis of the riddle. We therefore do not expect it to have 'surprising informational qualities', and this is indeed the case: i is a successful formula.

Formula i equals $\neg K_P(x,y)$ where $K_P(x,y)$ which was defined as $\bigvee_{(i,j)\in I} K_P(x_i \wedge y_j)$. Therefore, it has form $\neg K_n\varphi \wedge \neg K_n\psi \wedge \ldots$, with $\varphi, \psi, \ldots$ booleans. We show that this formula is successful for two conjuncts, i.e., formula $[\neg K_n\varphi \wedge \neg K_n\psi](\neg K_n\varphi \wedge \neg K_n\psi)$ is valid for booleans φ and ψ; the case for the longer finite conjunction follows similarly.

Let M, w be arbitrary. Assume $M, w \models \neg K_n\varphi \wedge \neg K_n\psi$. We have to prove that $M|(\neg K_n\varphi \wedge \neg K_n\psi), w \models \neg K_n\varphi \wedge \neg K_n\psi$. From $M, w \models \neg K_n\varphi \wedge \neg K_n\psi$ follows that there are v and $v' \in \mathcal{D}(M)$ such that $v \sim_n w$ and $M, v \models \neg\varphi$, and $v' \sim_n w$ and $M, v' \models \neg\psi$, respectively. As $\sim_n$ is an equivalence relation, we also have that $v \sim_n v$ and $v \sim_n v'$, we have as well $M, v \models \neg K_n\varphi \wedge \neg K_n\psi$; similarly, $M, v' \models \neg K_n\varphi \wedge \neg K_n\psi$. In other words, both v and v' are in the domain of $M|(\neg K_n\varphi \wedge \neg K_n\psi)$. As the value of boolean propositions only depends on the current factual state, from $M, v \models \neg\varphi$ and $v \in \mathcal{D}(M|(\neg K_n\varphi \wedge \neg K_n\psi))$ follows $M|(\neg K_n\varphi \wedge \neg K_n\psi), v \models \neg\varphi$; and from the last follows $M|(\neg K_n\varphi \wedge \neg K_n\psi), w \models \neg K_n\varphi$. Similarly, $M|(\neg K_n\varphi \wedge \neg K_n\psi), v' \models \neg\psi$; from which follows $M|(\neg K_n\varphi \wedge \neg K_n\psi), w \models \neg K_n\psi$. Therefore $M|(\neg K_n\varphi \wedge \neg K_n\psi), w \models \neg K_n\varphi \wedge \neg K_n\psi$, as required.

The case ii An agent can become ignorant from professing his own knowledge, and announcement *ii* is a typical example. This may sound strange[3], but it can easily be observed to be true for announcement *ii*: after *ii*, Product knows the numbers (formula *iii*), so it can no longer be true that Sum knows that Product does not know the numbers: in other words, formula *ii* is now false. Ergo, *ii* is an unsuccessful update.

The cases iii and iv The last two announcements *iii* and *iv* are successful formulas: this is because they are *preserved* formulas: they are truth preserving under submodel restrictions, an inductively defined fragment with – among other clauses – inductive clauses that atomic propositions are always preserved, and that if φ and ψ are preserved, then also $\varphi \wedge \psi$, $\varphi \vee \psi$, and $K_n\varphi$ [vB02]. The announcements *iii* and *iv* are disjunctions of formulas of the form $K_n(x_i \wedge y_j)$, and are therefore preserved. All preserved formulas are successful [vDK05]. And all successful formulas induce successful updates in all epistemic states.

No inductive definition of the successful formulas is known – in particular, if φ and ψ are both successful, $[\varphi]\psi$ may be unsuccessful. Having said that, it is remarkable that the sequence of the three announcements *ii* ; *iii* ; *iv* is an unsuccessful update, or, put in a single formula: $ii \wedge [ii]iii \wedge [ii \wedge [ii]iii]iv$ is unsuccessful in the initial epistemic state. This formula becomes false after its announcement: after that, just like after *ii*, Sum knows that Product knows the numbers, so it is now false that Sum knows that Product does not know the numbers: *ii* has become false, and therefore the entire conjunction corresponding to the sequence *ii* ; *iii* ; *iv*. The first announcement *i* can also be added to

[3]Stranger even, is that an agent can also become knowledgeable from professing his own ignorance, for which there are other examples.

the conjunction, so that $i \wedge ii \wedge [ii]iii \wedge [ii \wedge [ii]iii]iv$ is also unsuccessful in the initial epistemic state.

This last observation captures, we think, more than anything else our intuition that the Sum-and-Product problem is puzzling.

7 The Epistemic Model Checker DEMO

Recently, epistemic model checkers have been developed to verify properties of interpreted systems, knowledge-based protocols, and various other multi-agent systems. The model checkers MCK [GvdM04] and MCMAS [RL04] use the interpreted system architecture; MCK does this in a setting of linear and branching time temporal logic. The exploration of the search space in both MCK and MCMAS is based on ordered binary decision diagrams.

A different model checker, not based on a temporal epistemic architecture, is DEMO. It has been developed by Jan van Eijck [vE04]. DEMO is short for Dynamic Epistemic MOdelling. It allows modelling epistemic updates, graphical display of Kripke structures involved, and formula evaluation in epistemic states. DEMO is written in the functional programming language Haskell.

The model checker DEMO implements the dynamic epistemic logic of [BM04]. In this 'action model logic' the global state of a multi-agent system is represented by an epistemic model as in Section 3. But more epistemic actions are allowed than just public announcements, and each epistemic action is represented by an *action model*. Just like an epistemic model, an action model is also based on a multi-agent Kripke frame, but instead of carrying a valuation it has a precondition function that assigns a precondition to each point in the action model. A point in the action model domain stands for an atomic action. The epistemic state change in the system is via an operation called the *update product*. This is a restricted modal product. In this submission we restrict our attention to action models for public announcements. Such action models have a singleton domain, and the precondition of that point is the announced formula. We refrain from details and proceed with (a relevant part of – recursive clauses describing the effect of updates have been omitted) the recursive definition of formulas in DEMO.

```
Form = Top | Prop Prop | Neg  Form | Conj [Form] | Disj [Form]
          | K Agent Form | CK [Agent] Form
```

Formula `Top` stands for $\top$, `Prop Prop` for atomic propositional letters (the first occurrence of `Prop` means that the datatype is 'propositional atom', whereas the second occurrence of `Prop` is the placeholder for an actual proposition letter, such as `P 3`), `Neg` for negation, `Conj [Form]` stands for the conjunction of a list of formulas of type `Form`, similarly for `Disj`, `K Agent` stands for the individual knowledge operator for agent `Agent`, and `CK [Agent]` for the common knowledge operator for the group of agents listed in `[Agent]`.

The pointed and singleton action model for a public announcement is created by a function `public` with a precondition (the announced formula) as argument. The update operation is specified as

```
upd ::  EpistM -> PoAM -> EpistM
```

Here, `EpistM` is an epistemic state and `PoAM` is a pointed action model, and the update generates a new epistemic state. If the input epistemic state `EpistM` corresponds to some (M, w), then in case of the truthful public announcement of φ the resulting `EpistM` has the form $(M|\varphi, w)$. We can also update with a list of pointed action models:

```
upds ::  EpistM -> [PoAM] -> EpistM
```

An example is the sequence of three announcements in the Sum-and-Product problem.

8 Sum and Product in DEMO

We implement the Sum-and-Product riddle in DEMO and show how the implementation finds the unique solution $(4, 13)$. Figure 2 contains the implementation.

A list is a standard data structure in Haskell, unlike a set. The set $I \equiv \{(x, y) \in \mathbb{N}^2 \mid 1 < x < y \text{ and } x + y \leq 100\}$ is realized in DEMO as the list

```
pairs = [(x,y)| x<-[2..100], y<-[2..100], x<y, x+y<=100]
```

Thus, $\{$ and $\}$ are replaced by $[$ and $]$, $\in$ is replaced by `<-`, and instead of I we name it `pairs`. A pair such as $(4, 18)$ is not a proper name for a domain element. In DEMO, natural numbers are such proper names. Therefore, we associate each element in `pairs` with a natural number and make a new list.

```
ipairs = zip [0..numpairs-1] pairs
```

Here, `numpairs` is the number of elements in `pairs`, and the function `zip` pairs the i-th element in `[0..numpairs-1]` with the i-th element in `pairs`, and makes that the i-th element of `ipairs`. For example, the first element in `ipairs` is `(0,(2,3))`.

The initial model of the Sum-and-Product riddle is represented as

```
msnp :: EpistM
msnp = (Pmod [0..numpairs-1] val acc [0..numpairs-1])
  where
  val = [(w,[P x, Q y]) | (w,(x,y))<- ipairs]
  acc = [(a,w,v)| (w,(x1,y1))<-ipairs, (v,(x2,y2))<-ipairs, x1+y1==x2+y2 ]++
        [(b,w,v)| (w,(x1,y1))<-ipairs, (v,(x2,y2))<-ipairs, x1*y1==x2*y2 ]
```

```
module SNP
where
import DEMO

pairs  = [(x,y)|x<-[2..100], y<-[2..100], x<y, x+y<=100]
numpairs = llength(pairs)
llength [] =0
llength (x:xs) = 1+ llength xs
ipairs = zip [0..numpairs-1] pairs

msnp :: EpistM
msnp = (Pmod [0..numpairs-1] val acc [0..numpairs-1])
  where
  val = [(w,[P x, Q y]) | (w,(x,y))<- ipairs]
  acc = [(a,w,v)| (w,(x1,y1))<-ipairs, (v,(x2,y2))<-ipairs, x1+y1==x2+y2 ]++
        [(b,w,v)| (w,(x1,y1))<-ipairs, (v,(x2,y2))<-ipairs, x1*y1==x2*y2 ]

fmrs1e = K a (Conj [Disj[Neg (Conj [Prop (P x),Prop (Q y)]),
         Neg (K b (Conj [Prop (P x),Prop (Q y)]))]| (x,y)<-pairs])
amrs1e = public (fmrs1e)
fmrp2e = Conj [(Disj[Neg (Conj [Prop (P x),Prop (Q y)]),
     K b (Conj [Prop (P x),Prop (Q y)]) ] )|(x,y)<-pairs]
amrp2e = public (fmrp2e)
fmrs3e = Conj [(Disj[Neg (Conj [Prop (P x),Prop (Q y)]),
     K a (Conj [Prop (P x),Prop (Q y)]) ] )|(x,y)<-pairs]
amrs3e = public (fmrs3e)

solution = showM (upds msnp [amrs1e, amrp2e, amrs3e])
```

Figure 2: The DEMO program `SNP.hs`. Comment lines have been removed.

Here, `msnp` is a multi-pointed epistemic model, that consists of a domain `[0..numpairs-1]`, a valuation function `val`, an accessibility relation function `acc`, and `[0..numpairs-1]` points. As the points of the model are the entire domain, we may think of this initial epistemic state as the (not-pointed) epistemic model underlying it.

The valuation function `val` maps each state in the domain to the subset of atoms that are true in that state. This is different from Section 3, where the valuation V was defined as a function mapping each atom to the set of states where it is true. The correspondence $q \in \mathtt{val}(w)$ iff $w \in V(q)$ is elementary. An element `(w,[P x, Q y])` in `val` means that in state `w`, atoms `P x` and `Q y` are true. For example, given that `(0,(2,3))` is in `ipairs`, `P 2` and `Q 3` are true in state `0`, where `P 2` stands for 'the smaller number is 2' and `Q 3` stands for 'the larger number is 3'. These same facts were described in the previous section by x_2 and y_3, respectively, as that gave the closest match with the original problem formulation. In DEMO, names of atoms *must* start with capital P, Q, R, but the correspondence between names will be obvious.

The function acc specifies the accessibility relations. Agent a represents Sum and agent b represents Product. For (w,(x1,y1)) and (v,(x2,y2)) in ipairs, if their sum is the same: x1+y1==x2+y2, then they cannot be distinguished by Sum: (a,w,v) in acc; and if their product is the same: x1*y1==x2*y2, then they cannot be distinguished by Product: (b,w,v) in acc. Function ++ is an operation merging two lists.

Sum and Product's announcements are modelled as singleton action models, generated by the announced formula (precondition) φ and the operation public. Consider $K_S \neg \bigvee_{(i,j)\in I} K_P(x_i \wedge y_j)$, expressing that Sum says: "I knew you didn't." This is equivalent to $K_S \bigwedge_{(i,j)\in I} \neg K_P(x_i \wedge y_j)$. A conjunct $\neg K_P(x_i \wedge y_j)$ in that expression, for 'Product does not know that the pair is (i,j)', is equivalent to $(x_i \wedge y_j) \rightarrow \neg K_P(x_i \wedge y_j)$.[4] The latter is computationally cheaper to check in the model, than the former: in all states but (i,j) of the model, the latter requires a check on two booleans only, whereas the former requires a check *in each of those states* of Product's ignorance, that relates to his equivalence class for that state, and that typically consists of several states.

This explains that the check on $\bigwedge_{(i,j)\in I} \neg K_P(x_i \wedge y_j)$ can be replaced by one on $\bigwedge_{(i,j)\in I}((x_i \wedge y_j) \rightarrow \neg K_P(x_i \wedge y_j))$. Using a model validity, the check on $\bigvee_{(i,j)\in I} K_P(x_i \wedge y_j)$ (Product knows the numbers) can also be replaced, namely by a check $\bigwedge_{(i,j)\in I}((x_i \wedge y_j) \rightarrow K_P(x_i \wedge y_j))$.[5] Using these observations, and writing an implication $\varphi \rightarrow \psi$ as $\neg\varphi \vee \psi$, the three problem announcements ii, iii, and iv listed on page 11 are checked in DEMO by the formulas fmrs1e, fmrp2e, and fmrs3e, respectively, as listed in Figure 2. The corresponding singleton action models are obtained by applying the function public, namely as amrs1e = public (fmrs1e), amrp2e = public (fmrp2e), and amrs3e = public (fmrs3e). This is also shown in the figure.

Finally, we show a relevant part of DEMO interaction with this implementation. The complete (three-page) output of this interaction can be found on www.cs.otago.ac.nz/staffpriv/hans/sumpro/.

The riddle is solved by updating the initial model msnp with the action models corresponding to the three successive announcements:

```
*SNP> showM (upds msnp [amrs1e, amrp2e, amrs3e])
==> [0]
[0]
(0,[p4,q13])
(a,[[0]])
(b,[[0]])
```

This function showM displays a pointed epistemic model as:

```
==> [<points>]
```

[4] We use the $S5$-validity $\neg K\varphi \leftrightarrow (\varphi \rightarrow \neg K\varphi)$, that can be shown as follows: $\neg K\varphi$ iff $(\varphi \vee \neg\varphi) \rightarrow \neg K\varphi$ iff $(\varphi \rightarrow \neg K\varphi) \wedge (\neg\varphi \rightarrow \neg K\varphi)$ iff $(\varphi \rightarrow \neg K\varphi) \wedge (K\varphi \rightarrow \varphi)$ iff (in $S5$!) $(\varphi \rightarrow \neg K\varphi)$.

[5] We now use that $\varphi \overline{\vee} \psi$ – where $\overline{\vee}$ is exclusive disjunction – entails that ($K\varphi \vee K\psi$ iff $(\varphi \rightarrow K\varphi) \wedge (\psi \rightarrow K\psi)$).

```
[<domain>]
[<valuation>]
[<accessibility relations represented as equivalence classes>]
```

The list [p4,q13] represents the facts P 4 and Q 13, i.e., the solution pair $(4, 13)$. Sum and Product have full knowledge (their access is the identity) on this singleton domain consisting of state 0. That this state is named 0 is not a coincidence: after each update, states are renumbered starting from 0.

For another example, (upds msnp [amrs1e,amrp2e]) represents the model that results from Product's announcement (*iii*) "Now I know the numbers." Part of the showM results for that model are

```
*SNP> showM (upds msnp [amrs1e,amrp2e])
==> [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,
(...)
(0,[p2,q9])(1,[p2,q25])(2,[p2,q27])(3,[p3,q8])(4,[p3,q32])
(5,[p3,q38])(6,[p4,q7])(7,[p4,q13])(8,[p4,q19])(9,[p4,q23])
(...)
(a,[[0,3,6],[1,9,14,23,27,32,37,44,50],[2,10,17,24,28,38,45,46,51],[4
,11,18,29,33,39,47,55,60,65],[5,12,25,35,41,48,52,56,57,62,67,70,73],
[7],[8,22,36],[13,20,26,42,53,58,63,68,71,74,76,79,81],[15,19,30,34,4
0,61,66],[16,21,31,43,49,54,59,64,69,72,75,77,78,80,82,83,84,85]])
(b,[[0],[1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12],[13],[14],
(...)
```

After two announcements 86 pairs (x, y) remain possible. All remaining states are renumbered, from 0 to 85, of which part is shown. Product's (b) access consists of singleton sets only, of which part is shown. That should be obvious, as he just announced that he knew the number pair. Sum's (b) equivalence class [0,3,6] is that for sum 11: note that (0,[p2,q9]), (3,[p3,q8]), and (6,[p4,q7]) occur in the shown part of the valuation. Sum's access has one singleton equivalence class, namely [7]. That corresponds to the state for pair $(4, 13)$: see (7,[p4,q13]) in the valuation. Therefore, Sum can now truthfully announce to know the pair of numbers, after which the singleton final epistemic state (that was already displayed) results.

9 Other model checkers and DEMO programs

As mentioned in the introduction to the previous section, other model checkers around are MCK [GvdM04], and MCMAS [RL04]. The question is whether we could also implement this problem in those model checkers. For the latest versions of these model checkers in both cases the answer appears to be 'no'.

The current version of MCK is 0.2.0. In MCK, a state of the environment is an assignment to a set of variables declared in the environment section. These variables are usually assumed to be partially accessible to the individual agents, and agents could share some variables. The change of the state of the multi-agent system is either made by agents or the environment, in the form

of changing these variables. There are two ways to make such changes. One is to send signals to the environment using the action construct by agents in conjunction with the transitions construct by the environment, which provides a way to describe how the environment variables are updated. The other is a specialized form for actions from the perspective that environment variables are shared variables, by providing read and write operations on those shared variables. In both cases, we need guarded statements to make the change. For example, a simple deterministic statement has the form:

$$\text{if } cond \rightarrow C \text{ [otherwise} \rightarrow C_o \text{] fi}$$

where command C is eligible for execution only if the corresponding condition *cond* evaluates to true in the current state. Otherwise, the command C_o will be executed. If we would like to model the Sum-and-Product problem in MCK, the effect of a public announcement should be recorded in a variable which is accessible to all agents. Suppose the effect of P's public announcement : "I now know it" ($K_P(x, y)$) is recorded in variable v. Then in a state just after this announcement, the variable v will be set to $True$ if $K_P(x, y)$ holds in the previous state, and otherwise to $False$. Clearly, we need that statement in the above *epistemic* form, with *cond* involving knowledge checking. Unfortunately, even though in MCK we can check epistemic postconditions, the current version of MCK does not support checking epistemic formulas as preconditions, as in *cond*. This might possibly be related to inherent difficulties to incorporate knowledge in *cond*, but an extension seems called for.

The latest MCMAS is version 0.7. The underlying theory has been developed by Alessio Lomuscio. It can be seen as a continuation of his PhD work on hypercube systems, which are a special class of interpreted systems [Lom99]. Similarly to MCK, MCMAS also does not support actions with knowledge-based preconditions to transit from one global state to another global state.

Apart from the Sum-and-Product riddle we have implemented some of the other riddles discussed in this paper in DEMO, such as the 'Domiciliary' problem in Section 2.2. All these programs are mere variations of the one presented in this paper, because the communications are always similar public announcements of knowledge and ignorance in a two-agent system, whereas only the 'starting conditions' – the number (or symbol) pairs initially allowed – vary from problem to problem. For these programs, including full explanations, we refer to the website `www.cs.otago.ac.nz/staffpriv/hans/sumpro/`.

10 Conclusions

We have modelled the Sum-and-Product problem in public announcement logic and verified its properties in the epistemic model checker DEMO. The problem can be represented in the traditional way by number pairs, so that Sum knows their sum and Product their product, but also as an interpreted system with (sum,product) pairs. Subject to the union of languages, the representations are

bisimilar, and even isomorphic. We also analyzed which announcements made towards a solution of the problem were unsuccessful updates – formulas that become false because they are announced.

A final word on model checking such problems: originally, an analysis involving elementary number theory and combinatorics was necessary to solve the problem. Indeed, that was the whole fun of the problem. Solving it in a model checker instead, wherein one can, in a way, simply state the problem in its original epistemic formulation, hides all that combinatorial structure and makes it appear almost trivial. Far from trying to show that the problem is therefore actually trivial or uninteresting, this rather shows how powerful model checking tools may be, when knowledge specifications are clear and simple but their structural ramifications complex.

References

[BdRV01] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, Cambridge, 2001. Cambridge Tracts in Theoretical Computer Science 53.

[BM96] J. Barwise and L.S. Moss. *Vicious Circles*. CSLI Publications, Stanford, 1996.

[BM04] A. Baltag and L.S. Moss. Logics for epistemic programs. *Synthese*, 139:165–224, 2004. Knowledge, Rationality & Action 1–60.

[BMS98] A. Baltag, L.S. Moss, and S. Solecki. The logic of common knowledge, public announcements, and private suspicions. In I. Gilboa, editor, *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 98)*, pages 43–56, 1998.

[FHMV95] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge MA, 1995.

[Fre69] H. Freudenthal. (formulation of the sum-and-product problem). *Nieuw Archief voor Wiskunde*, 3(17):152, 1969.

[Fre70] H. Freudenthal. (solution of the sum-and-product problem). *Nieuw Archief voor Wiskunde*, 3(18):102–106, 1970.

[Gar79] M. Gardner. Mathematical games. *Scientific American*, 241 (December):20–24, 1979. Also addressed in the March (page 24) and May (pages 20–21) issues of volume 242, 1980.

[Ger99] J.D. Gerbrandy. *Bisimulations on Planet Kripke*. PhD thesis, University of Amsterdam, 1999. ILLC Dissertation Series DS-1999-01.

[Ger05] J.D. Gerbrandy. The surprise examination in dynamic epistemic logic. *Synthese*, 2005. To appear.

[Gre68] M. H. Greenblatt. *Mathematical Entertainments: A Collection of Illuminating Puzzles New and Old*. George Allen and Unwin Ltd, London, 1968.

[GvdM04] P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In R. Alur and D. Peled, editors, *Proceedings of the 16th International Conference on Computer Aided Verification (CAV 2004)*, pages 479–483. Springer, 2004.

[Hin62] J. Hintikka. *Knowledge and Belief*. Cornell University Press, Ithaca, NY, 1962.

[Isa95] I.M. Isaacs. The impossible problem revisited again. *The Mathematical Intelligencer*, 17(4):4–6, 1995.

[Koo05] B.P. Kooi. On public update logics. Under submission, 2005.

[Lew69] D.K. Lewis. *Convention, a Philosophical Study*. Harvard University Press, Cambridge (MA), 1969.

[Liu04] A. Liu. Problem section: Problem 182. *Math Horizons*, 11:324, 2004.

[Lom99] A.R. Lomuscio. *Knowledge Sharing among Ideal Agents*. PhD thesis, University of Birmingham, Birmingham, UK, 1999.

[McC90] J. McCarthy. Formalization of two puzzles involving knowledge. In Vladimir Lifschitz, editor, *Formalizing Common Sense : Papers by John McCarthy*, Ablex Series in Artificial Intelligence. Ablex Publishing Corporation, Norwood, N.J., 1990. original manuscript dated 1978–1981.

[MDH86] Y. O. Moses, D. Dolev, and J. Y. Halpern. Cheating husbands and other stories: a case study in knowledge, action, and communication. *Distributed Computing*, 1(3):167–176, 1986.

[Moo42] G.E. Moore. A reply to my critics. In P.A. Schilpp, editor, *The Philosophy of G.E. Moore*, pages 535–677. Northwestern University, Evanston IL, 1942. The Library of Living Philosophers (volume 4).

[Pan91] G. Panti. Solution of a number theoretic problem involving knowledge. *International Journal of Foundations of Computer Science*, 2(4):419–424, 1991.

[Pla89] J.A. Plaza. Logics of public communications. In M.L. Emrich, M.S. Pfeifer, M. Hadzikadic, and Z.W. Ras, editors, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*, pages 201–216, 1989.

[RL04] Franco Raimondi and Alessio Lomuscio. Verification of multiagent systems via ordered binary decision diagrams: An algorithm and its implementation. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 04)*, pages 630–637. IEEE Computer Society, 2004.

[Sal95] L. Sallows. The impossible problem. *The Mathematical Intelligencer*, 17(1):27–33, 1995.

[Sch72] Stephen Schiffer. *Meaning*. Oxford University Press, Oxford, 1972.

[vB98] J.F.A.K. van Benthem. Dynamic odds and ends. Technical report, ILLC, University of Amsterdam, 1998. Report ML-1998-08.

[vB02] J.F.A.K. van Benthem. One is a lonely number: on the logic of communication. Technical report, ILLC, University of Amsterdam, 2002. Report PP-2002-27 (material presented at the Logic Colloquium 2002).

[vdHV02] W. van der Hoek and L.C. Verbrugge. Epistemic logic: a survey. In L.A. Petrosjan and V.V. Mazalov, editors, *Game theory and Applications*, volume 8, pages 53–94, 2002.

[vDK05] H.P. van Ditmarsch and B.P. Kooi. The secret of my success. *Synthese*, 2005. To appear.

[vdM94] R. van der Meyden. Mutual belief revision. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 595–606. Morgan Kaufmann, 1994.

[vDRV05] H.P. van Ditmarsch, J. Ruan, and L.C. Verbrugge. Model checking sum and product. In *Proceedings of the 18th Australian Joint Conference on Artificial Intelligence (AI 2005)*, pages 790–795. Springer Verlag, 2005. LNAI 3809.

[vDvdHK03] H.P. van Ditmarsch, W. van der Hoek, and B.P. Kooi. Descriptions of game states. In G. Mints and R. Muskens, editors, *Logic, Games, and Constructive Sets*, pages 43–58, Stanford, 2003. CSLI Publications. CSLI Lecture Notes No. 161.

[vDvdHK05] H.P. van Ditmarsch, W. van der Hoek, and B.P. Kooi. Dynamic epistemic logic. Manuscript, 2005.

[vE04] J. van Eijck. Dynamic epistemic modelling. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam, 2004. CWI Report SEN-E0424.

[WS40] W. T. Williams and G. H. Savage. *The Penguin Problems Book: A Modern Anthology of Perplexities and Tantalizers*. Penguin Books (Allen Lane), Harmondsworth, 1940.

[WS44] W. T. Williams and G. H. Savage. *The Second Penguin Problems Book*. Penguin Books, Harmondsworth, 1944.